<Translation of Japanese Patent Application No. HEI 11-138649>

[Name of Document]	Application for Patent
[Docket Number]	9805670
[Filing Date]	May 19, 1999
[To]	Commissioner, Patent Office
[International Patent Classification]	G06F 17/50 F16H 21/00
[Title of Invention]	Mechanism-Control-Program Development Support System, Mechanism-Control-Program Development Support Device, and Mechanism-Control-Program Development Support Program Storage Medium
[Number of Claims]	3

[Inventor]
[Address or Domicile] c/o FUJITSU LIMITED, 1-1, Kamikodanaka 4-chome, Nakahara-ku, Kawasaki-shi, Kanagawa
[Name] Yosuke SENTA

[Inventor]
[Address or Domicile] c/o FUJITSU LIMITED, 1-1, Kamikodanaka 4-chome, Nakahara-ku, Kawasaki-shi, Kanagawa
[Name] Yuici SATO

[Inventor]
[Address or Domicile] c/o FUJITSU LIMITED, 1-1, Kamikodanaka 4-chome, Nakahara-ku, Kawasaki-shi, Kanagawa
[Name] Mitsunori HIRATA

Cont'd....

[Applicant for Patent]
[Identification Number] 000005223
[Name] FUJITSU LIMITED

[Agent]
[Identification Number] 100094330
[Patent Attorney]
[Name] Masaki YAMADA

[Priority Claimed]
[International Patent Application No.] PCT/JP/98/05685
[Filing Date] December 16, 1998

[Indication of Official Fees]
[Advance Deposit Record Number] 017961
[Amount Paid] ¥21,000

[List of Materials Submitted]

[Name of Material]	Specification	1
[Name of Material]	Drawings	1
[Name of Material]	Abstract	1
[Number of General Power of Attorney]		9704376

[Whether a proof is required] Required

[Tittle of Document] Specification

[Title of the Invention]

Mechanism-Control-Program Development Support System, Mechanism-Control-Program Development Support Device, and Mechanism-Control-Program Development Support Program Storage Medium

[Scope of Claims for a Patent]

[Claim 1]

A mechanism-control-program development support system comprising:

a three-dimensional mechanism model simulator for operating a three-dimensional mechanism model, constructed within said simulator, which is composed of a plurality of parts including an actuator and a sensor; and

a control program processor for executing a control program for controlling operation of said three-dimensional mechanism model constructed within said three-dimensional mechanism model simulator, while synchronizing said control program with said three-dimensional mechanism model which operates within said three-dimensional mechanism model simulator.

[Claim 2]

A mechanism-control-program development support device for executing a control program, for controlling operation of a three-dimensional mechanism model constructed within a

three-dimensional mechanism model simulator for operating said three-dimensional mechanism model, while synchronizing said control program with said three-dimensional mechanism model which operates within said three-dimensional mechanism model simulator,

said three-dimensional mechanism model being composed of a plurality of parts including an actuator and a sensor.

[Claim 3]

A mechanism-control-program development support program storage medium in which a mechanism-control-program development support program is stored, said mechanism-control-program development support program having:

a three-dimensional mechanism model simulation program for operating a three-dimensional mechanism model which is composed of a plurality of parts including an actuator and a sensor; and

a synchronizing program, added to a control program for controlling the operation of said three-dimensional mechanism model, for executing said control program while synchronizing said control program with the operation of said three-dimensional mechanism model.

[Detailed Description of the Invention]

[0001]

[Technical Field Pertinent to the Invention]

The present invention relates to a mechanism-control-program development support system and

mechanism-control-program development support device for supporting the development of a control program for controlling a mechanism, and also relates to a mechanism-control-program development support program storage medium in which a mechanism-control-program development support program for supporting the development of the control program for controlling a mechanism is stored.

[0002]

[Prior Art]

Recently, with the advancement of computer graphics and simulation techniques, a three-dimensional mechanism model can be constructed and the mechanism can be confirmed by simulation without manufacturing a trial product. For example, by moving a moving part of the three-dimensional mechanism model, it can be confirmed whether parts can perform a desired operation without contacting each other unexpectedly. As a result, it becomes possible to detect a defect in design in its early stages and also to change disposition of parts or perform modification including moving parts. This makes it possible to reduce the time and cost required for product development.

[0003]

To operate such a mechanism, programs for controlling such a mechanism have been developed. The mechanism is usually operated by executing the control program.

[0004]

However, to develop such a control program, a nearly finished mechanism must be physically manufactured on an

experimental basis, for example, by a simulation of operation for a three-dimensional mechanism model. The control program is developed, while the manufactured mechanism is being operated. Thus, since the development and debugging of the control program require a trial product, the control program cannot be developed during the time that the trial product is finished, or during the time that the trial product is repaired or improved. Thus, the development of the control program is fairly time-consuming and costly.

[0005]

[Problems to be solved by the Invention]

In view of the aforementioned circumstances, it is an object of the present invention to provide a mechanism-control-program development support system and a mechanism-control-program development support device which are capable of facilitating the development of a program for controlling a mechanism. Another object of the invention is to provide a mechanism-control-program development support program storage medium in which a mechanism-control-program development support program for facilitating the development of a program for controlling a mechanism is stored.

[0006]

[Means for Solving the Problems]

To achieve the aforementioned object of the invention, there is provided a mechanism-control-program development support system comprising:

a three-dimensional mechanism model simulator for

operating a three-dimensional mechanism model, constructed within the simulator, which is composed of a plurality of parts including an actuator and a sensor; and

a control program processor for executing a control program for controlling operation of the three-dimensional mechanism model constructed within the three-dimensional mechanism model simulator, while synchronizing the control program with the three-dimensional mechanism model which operates within said three-dimensional mechanism model simulator.

[0007]

In the case of constructing the three-dimensional mechanism model and operating it, the operating speed differs largely from that in the case of manufacturing a trial product equivalent to the three-dimensional mechanism model, and operating the trial product. Therefore, even if the control program is executed to operate the three-dimensional mechanism model, it will be unsuitable for debugging the control program. On the other hand, the control program processor, which constitutes the mechanism-control-program development support system of the present invention, executes the control program in synchronization with operation of the three-dimensional mechanism model that is controlled by the control program. Consequently, the control program can be confirmed by operating the three-dimensional mechanism model under the control of the control program, and confirming the operation.

[0008]

Thus, according to the mechanism-control-program development support system of the present invention, an operating program can be debugged while operating the three-dimensional mechanism model by simulation, without manufacturing a trial product. This achieves a substantial reduction in the time and cost required for the development of an operating program.

[0009]

Here, in the mechanism-control-program development support system of the present invention, operation of the three-dimensional mechanism model and execution of the control program may be synchronized as follows:

(1) The three-dimensional mechanism model simulator notifies the control program processor of an execution time for a control program that operates within the control program processor;

(2) In response to the notification of the execution time for the control program from the three-dimensional mechanism model simulator, the control program processor executes the control program by the execution time and notifies the termination of the execution of the control program for the execution time to the three-dimensional mechanism model simulator; and

(3) In response to the notification of the termination of the execution of the control program for the execution time from the control program processor, the three-dimensional mechanism model simulator operates the three-dimensional

mechanism model by an amount of operation corresponding to the control of the control program for the execution time, and determines the next execution time.

[0010]

For example, with procedure such as this, operation of the three-dimensional mechanism model can be synchronized with execution of the control program.

[0011]

Here, it is preferable that for obtaining synchronization by adopting the aforementioned procedure, the three-dimensional mechanism model simulator determine the execution time for the control program, which operates within the control program processor, in accordance with a distance between parts constituting the three-dimensional mechanism model.

For instance, when parts are far away from each other, the time until a desired entire operation is performed can be shortened by increasing the execution time and largely moving one of the parts once. On the other hand, when parts approach each other, accurate operation can be confirmed by setting the execution time to a small value and moving one of the parts little by little.

[0012]

In a preferred form of the present invention, for obtaining a distance between parts, the three-dimensional mechanism model simulator divides a plurality of parts, constituting a three-dimensional mechanism model, into groups for each

actuator constituting the three-dimensional mechanism model, prior to the operation of the three-dimensional mechanism model synchronized with the execution of the control program. More specifically, the plurality of parts are separated into a group of parts that move according to operation of the actuator and a group of parts that stay in a stopped state, depending on operation of the actuator. For the operation of the three-dimensional mechanism model synchronized with execution of the operating program, it is preferable to obtain a distance between parts in these separated groups, except a distance between parts in an interfering state between these groups.

[0013]

As described above, by dividing parts into groups and obtaining a distance between the groups, the calculation time to obtain the distance can be greatly shortened.

[0014]

In another preferred form of the present invention, the three-dimensional mechanism model simulator extracts a group of parts which move according to operation of the actuator constituting the three-dimensional mechanism model, from a plurality of parts constituting the three-dimensional mechanism model, for each actuator, prior to the operation of the three-dimensional mechanism model synchronized with the execution of the control program. The actuator is operated to move the parts constituting the group, and interference between the parts is detected. For the operation of the three-dimensional mechanism model synchronized with execution

of the control program, a distance between interfering parts in an interfering state is obtained. However, parts that are always in the interfering state are excluded.

[0015]

There is a possibility that when parts are divided into groups, as described above, parts will interfere with each other even in the group of parts that move according to operation of the actuator. In such a case, parts constituting the group are moved and interference between the parts is detected. If a distance between the interfering parts is obtained, a distance between parts having the possibility of interference can be obtained without an oversight.

[0016]

In still another preferred form of the present invention, the three-dimensional mechanism model simulator obtains a distance between parts constituting the three-dimensional mechanism model, only when an actuator constituting the three-dimensional mechanism model is being operated.

[0017]

High-speed simulation can be achieved by calculating a distance between parts only when needed.

[0018]

When synchronizing the three-dimensional mechanism model, which opertes within the three-dimensional mechanism model simulator, with the operating program that is executed within the control program processor, it is preferable that the control program, which is executed within the control program processor,

have a section to be repeatedly executed in predetermined cycles at least one portion, and recognize the termination of the execution of the control program for the execution time, based on the number of executions of the section.

[0019]

Generally, the control program has a section to be repeatedly executed in predetermined cycles. For example, the section is executed for each fixed time by timer interruption. In such a case, the execution of the control program for the execution time can be recognized by counting the number of executions of the section.

[0020]

The mechanism-control-program development support system of the present invention is equipped with actuator definition means for defining an actuator, constituting the three-dimensional mechanism model, which operates within the three-dimensional mechanism model simulator. Preferably, in defining the actuator, the actuator definition means is provided with two definition menus, that is, a stepped speed change for defining a speed instruction value and a target speed at 1:1 and an infinite speed change for defining a proportional constant between the speed instruction value and the target speed.

[0021]

By adopting the two methods of definition, the degree of freedom for actuation definition is enhanced. While the stepped speed change and the infinite speed change have been described, the stepped speed change or infinite speed change

may be divided into a plurality of definition methods. In an embodiment to be described later, the infinite speed change is divided into two kinds of definition methods.

[0022]

In the mechanism-control-program development support system of the present invention, it is preferable that the three-dimensional mechanism model simulator operate an actuator, constituting the three-dimensional mechanism model, which operates within the three-dimensional mechanism model simulator, as a primary delay system.

[0023]

By operating the actuator as a primary delay system, the behavior of the rise time of the actuator can be specified only by specifying a settling time which represents a delay time when operation of the actuator reaches approximately a constant operating speed. Thus, operation of the actuator can be easily defined.

[0024]

The mechanism-control-program development support system of the present invention is further equipped with actuator definition means for defining an actuator, constituting the three-dimensional mechanism model, which operates within the three-dimensional mechanism model simulator. It is preferable that the actuator definition means define a part, which first operates by operation of the actuator, as an actuator in the case of operating the three-dimensional mechanism model within the three-dimensional mechanism model simulator.

[0025]

By defining the actuator in this manner, a simulation of operation for the three-dimensional mechanism model becomes easy.

[0026]

The mechanism-control-program development support system of the present invention is further equipped with sensor definition means for defining a sensor, constituting the three-dimensional mechanism model, which operates within the three-dimensional mechanism model simulator. It is preferable that the sensor definition means define a part, detected by the sensor, as a sensor in the case of operating the three-dimensional mechanism model within the three-dimensional mechanism model simulator.

[0027]

By defining the sensor in this manner, as with the definition of the actuator, simulation becomes easy.

[0028]

The mechanism-control-program development support system of the present invention is further equipped with sensor definition means for defining a sensor, constituting the three-dimensional mechanism model, which operates within the three-dimensional mechanism model simulator. The sensor definition means may define a sensor for judging an ON state and an OFF state by detecting interference between a plurality of parts.

[0029]

The sensor is able to facilitate simulation even by such a definition method.

The mechanism-control-program development support system of the present invention is further equipped with encoder definition means for defining an encoder, constituting the three-dimensional mechanism model, which operates within the three-dimensional mechanism model simulator. Preferably, the three-dimensional mechanism model simulator receives the notification of the termination of the execution of the control program for the aforementioned execution time from the control program sensor and, at the stage that the three-dimensional mechanism model has been operated by an amount of operation which corresponds to the control of the control program for the execution time, outputs a pulse signal having the number of pulses which corresponds to the amount of operation of the encoder corresponding to the control for the execution time.

[0030]

The encoder can be accurately simulated by defining the encoder and outputting a pulse signal which has the number of pulses which corresponds to the amount of operation of the encoder corresponding to the control for the execution time, at the stage that the encoder has been operated by an amount of control for a single execution time.

[0031]

Preferably, the encoder definition means has the function of setting the upper limit of the pulse frequency of the output pulse signal of the encoder and, when outputting a pulse signal

having the number of pulses corresponding to the amount of operation of the encoder, outputs a pulse signal which has the number of pulses less than the upper limit of the pulse frequency.

[0032]

The output pulse signal of an actual encoder has a frequency less than a certain constant pulse frequency. Therefore, more accurate simulation becomes possible by setting the upper limit of a pulse frequency and outputting a pulse signal which has a pulse frequency less than the upper limit.

[0033]

Preferably, the mechanism-control-program development support system of the present invention is equipped with a potentiometer definition means for defining a potentiometer, constituting the three-dimensional mechanism model, which operates within the three-dimensional mechanism model simulator.

[0034]

The reason for that is that the potentiometer can also be an important constituent element of the three-dimensional mechanism model.

[0035]

Preferably, the mechanism-control-program development support system of the present invention is further equipped with sensor definition means for defining a sensor, constituting the three-dimensional mechanism model, which operates within the three-dimensional mechanism model simulator. The sensor definition means includes means for defining a delay between

the occurrence of the cause of changing a sensor state and an actual change in the sensor state.

[0036]

For development of the control program, it is preferable to hold an actuator or a sensor in a controlled state even when they operate abnormally. For development of such a control program, it is necessary to simulate abnormal operation of an actuator or a sensor.

If the mechanism-control-program development support system of the present invention is equipped with sensor definition means, and the sensor definition means includes means for defining a delay between the occurrence of the cause of changing a sensor state from the ON state (or OFF state) to the OFF state (or ON state) and an actual change in the sensor state, the delay can be defined so that an abnormal state such as a sensor response delay can be simulated.

[0037]

Preferably, the mechanism-control-program development support system of the present invention is further equipped with sensor definition means for defining a sensor, constituting the three-dimensional mechanism model, which operates within the three-dimensional mechanism model simulator. The sensor definition means includes means for defining chattering which results form a change in the sensor state.

[0038]

Chattering is also one of the abnormal operations that are often caused by the sensor, so it is important to simulate

the chattering.

[0039]

Preferably, the mechanism-control-program development support system of the present invention is equipped with sensor definition means for defining a sensor, constituting the three-dimensional mechanism model, which operates within the three-dimensional mechanism model simulator. The sensor definition means includes means for defining a sensor failure. Preferably, the mechanism-control-program development support system of the present invention is further equipped with actuator definition means for defining an actuator, constituting the three-dimensional mechanism model, which operates within the three-dimensional mechanism model simulator. The actuator definition means includes means for defining an actuator failure.

[0040]

It is also important to simulate a sensor failure or an actuator failure.

[0041]

Since the three-dimensional mechanism model is a mathematical model, two parts can be freely moved even if they overlap each other. However, there are cases where if actual parts interfere with each other, they cannot be moved any further. Therefore, in the mechanism-control-program development support system of the present invention, when interference occurs between parts during operation of the three-dimensional mechanism model, the above-mentioned three-dimensional

mechanism model simulator stops operation of the interfering parts.

[0042]

Preferably, the mechanism-control-program development support system of the present invention is equipped with defective-part definition means for specifying any one of parts, constituting the three-dimensional mechanism model, which operates within the three-dimensional mechanism model simulator, and inhibiting operation of the part.

[0043]

There are cases where in addition to sensors and actuators, ordinary driven parts will fail and operate incorrectly, so it is also important to simulate those parts.

[0044]

In that case, it is preferable that in response to the notification of a part specified by the above-mentioned defective-part definition means during operation of the three-dimensional mechanism model, the three-dimensional mechanism model simulator inhibit operation of the specified part in the current moving direction. It is also preferable that the above-mentioned three-dimensional mechanism model simulator release the inhibition of the operation of the part in the inhibited direction, performed by the above-mentioned defective-part definition means, in accordance with operation in the direction opposite to the operation-inhibited direction.

[0045]

In this manner, when a part is caught by something, a

state which returns to normalcy can be simulated by moving the caught part in the opposite direction.

[0046]

In the mechanism-control-program development support system of the present invention, it is preferable that the above-mentioned three-dimensional mechanism model simulator have a movable-range setting means for automatically setting the movable range of a part constituting the three-dimensional mechanism model, by detecting the interference between parts which occurs when the part is operated.

[0047]

By automatically setting the movable range of a part by the interference detecting method, a system is realized which eliminates the labor for the operator to input the movable range and is easy to handle and excellent in operability.

[0048]

In the mechanism-control-program development support system of the present invention, it is preferable that the three-dimensional mechanism model simulator have cam relation setting means for automatically setting a cam relation between two parts which constitute the three-dimensional mechanism model, by detecting the interference, between the two parts, which occurs when operating one of the two parts.

[0049]

For the interlocking operation between two parts which are in a cam relation, the cam relation is automatically set by the interference detecting method. This realizes a system

which eliminates operator's labor and is easy to handle.

[0050]

For the automatic setting of the cam relation, there are cases where one of two parts is urged in one direction. In that case, it is preferable that the cam relation setting means detect the interference between the two parts when the other of the two parts is urged in a predetermined direction, thereby automatically setting the cam relation therebetween.

[0051]

In the mechanism-control-program development support system of the present invention, it is preferable that the three-dimensional mechanism model simulator have groove relation setting means for automatically setting a groove relation that a groove in one of two parts constituting the three-dimensional mechanism model is inserted onto the other part (pin). The groove relation is automatically set by generating a virtual part smaller in diameter than the pin and detecting the distance between the virtual part and the groove wall.

[0052]

The pin inserted in the groove is usually in contact with the groove wall, so a non-contact state cannot be generated. Hence, the aforementioned small-diameter virtual part is generated and the distance between the virtual part and the groove wall is detected, whereby the groove relation can be automatically set.

[0053]

In the mechanism-control-program development support system of the present invention, it is preferable that the three-dimensional mechanism model simulator operate the three-dimensional mechanism model by recursively executing a program for operating a part which constitutes the three-dimensional mechanism model.

[0054]

It becomes possible to simulate operation of the three-dimensional mechanism model by adopting the program which is recursively executed.

[0055]

Preferably, the mechanism-control-program development support system of the present invention is equipped with clutch definition means for defining a clutch for switching the interlocking operation between two parts, constituting the three-dimensional mechanism model, which operate within the three-dimensional mechanism model simulator.

[0056]

Because the clutch can also be one of the important parts constituting the three-dimensional mechanism model, it is preferable to simulate the clutch.

[0057]

To achieve the aforementioned object of the invention, there is provided a mechanism-control-program development support device for executing a control program, for controlling operation of a three-dimensional mechanism model constructed within a three-dimensional mechanism model simulator for

operating the three-dimensional mechanism model, while synchronizing the control program with the three-dimensional mechanism model which operates within the three-dimensional mechanism model simulator,

the three-dimensional mechanism model being composed of a plurality of parts including an actuator and a sensor.

[0058]

When the three-dimensional mechanism model simulator is separately constructed, the simulator can support the development of an operating program by constituting a mechanism-control-program development support device.

[0059]

To achieve the aforementioned object of the invention, there is provided a mechanism-control-program development support program storage medium in which a mechanism-control-program development support program is stored, the mechanism-control-program development support program having:

a three-dimensional mechanism model simulation program for operating a three-dimensional mechanism model which is composed of a plurality of parts including an actuator and a sensor; and

a synchronizing program, added to a control program for controlling the operation of the three-dimensional mechanism model, for executing the control program while synchronizing the control program with the operation of the three-dimensional mechanism model.

[0060]

If the mechanism-control-program development support program, stored in the medium mechanism-control-program development support program storage medium of the present invention, is executed within a computer, an operating program can be debugged by simulation and therefore the development of the operating program will become easy.

[0061]

Preferably, in the mechanism-control-program development support program storage medium of the present invention,

the three-dimensional mechanism model simulator notifies an execution time for the control program to the synchronizing program;

in response to the notification of the execution time for the control program from the three-dimensional mechanism model simulator, the synchronizing program executes the control program by the execution time and notifies the termination of the execution of the control program for the execution time to the three-dimensional mechanism model simulation program; and

in response to the notification of the termination of the execution of the control program for the execution time from the control program processor, the three-dimensional mechanism model simulation program operates the three-dimensional mechanism model by an amount of operation corresponding to the control of the control program for the

execution time, and determines the next execution time.

[0062]

For example, with the aforementioned transmission of notification, the operation of the three-dimensional mechanism model can be synchronized with the execution of the control program.

[0063]

[Embodiments of the Invention]

Embodiments of the present invention will hereinafter be described.

[0064]

FIG. 1 is a diagram showing the exterior of a computer system in which a mechanism-control-program development support system is constructed.

[0065]

The computer system 100 comprises a main body portion 101 having incorporated a CPU, a RAM, a hard disk, etc.; a CRT display 102 for performing image display on a display screen 102a in response to an instruction from the main body portion 101; a keyboard 103 for inputting user's instruction and character information to this computer system; and a mouse 104 for inputting an instruction corresponding to an icon or the like, displayed at an arbitrary position on the display screen 102a by specifying the position.

[0066]

The main body portion 101 further has a floppy disk loading slot 101a and a CD-ROM loading slot 101b into which a floppy

disk 212 (not shown in FIG. 1; see FIG. 2) and a CD-ROM 210 are freely removably loaded. And inside the main body portion 101, a floppy disk driver 224 and a CD-ROM driver 225 (see FIG. 10) are also incorporated for driving the loaded floppy disk and CD-ROM 210.

[0067]

In this embodiment, a mechanism-control-program development support program according to the present invention is stored on the CD-ROM 210. This CD-ROM 210 is loaded into the main body portion 101 through the CD-ROM loading slot 101b, and the mechanism-control-program development support program stored on the CD-ROM 210 is installed in the hard disk of the computer system 100 by the CD-ROM driver 225. Furthermore, a three-dimensional mechanism model, defined with the keyboard, the mouse, etc. by the user, and a control program for controlling a product when actually manufacturing the three-dimensional mechanism model input with the keyboard, the mouse, etc. by the user, are stored on the hard disk of the computer system 100. Note that either the three-dimensional mechanism model or the control program or both may be constructed or programmed with another system separate from the computer 100 shown in FIG. 1, and may be stored in the computer 100.

[0068]

The three-dimensional mechanism model constructed within this computer system 100 is operated within the computer system 100 by a three-dimensional mechanism model simulation program which constitutes the mechanism-control-program development

support program loaded from the CD-ROM 210 into the computer system 100. On the other hand, an operating program incorporated into this computer system 100 is coupled with a synchronizing program which constitutes the mechanism-control-program development support program loaded from the CD-ROM 210 into the computer system 100. Within the computer system 100, the operating program is executed in synchronization with operation of the three-dimensional mechanism model.

[0069]

In this embodiment, the operating program is to be debugged. Therefore, although there is a possibility that the three-dimensional mechanism model itself constructed within the computer system 100 will be corrected when a defect is found in the course of cooperation with the operating program, the operation of the three-dimensional mechanism model has beforehand been confirmed and the operating program is a program being debugged.

[0070]

In connection with this embodiment of the present invention, the function of the computer system 100 of operating the three-dimensional mechanism model corresponds to the three-dimensional mechanism model stimulator of the present invention, while the function of executing the operating program coupled with the synchronizing program corresponds to the control program processor and the mechanism-control-program development support device of the present invention.

[0071]

Thus, the mechanism-control-program development support system of the present invention also includes a case where both the three-dimensional mechanism model simulator and the operating-program processor are incorporated into a single system (in this embodiment, the computer system 100).

[0072]

FIG. 2 is a block diagram showing the hardware of the computer system shown in FIG. 1.

[0073]

In FIG. 2 there are shown a central processing unit (CPU) 221, a RAM 222, a hard disk controller 223, a floppy disk driver 224, a CD-ROM driver 225, a mouse controller 226, a keyboard controller 227, a display controller 228. They are interconnected via a bus 220.

[0074]

The floppy disk driver 224 and the CD-ROM driver 225 are loaded with the floppy disk 212 and the CD-ROM 210, as described with reference to FIG. 1, and are used for accessing the loaded floppy disk 212 and CD-ROM 210.

[0075]

FIG. 2 also shows the hard disk 211 to be controlled by the hard disk controller 223, the mouse 104 to be controlled by the mouse controller 226, the keyboard 103 to be controlled by the keyboard controller 227, and the CRT display 102 to be controlled by the display controller 228.

[0076]

As described above, the mechanism-control-program development support program is stored on the CD-ROM 210. The mechanism-control-program development support program is read out from the CD-ROM 210 by the CD-ROM driver 225 and is stored in the hard disk 211 via the bus 220 by the hard disk controller 223. Practically, the program in the hard disk 211 is loaded onto the RAM 222 and is executed by the CPU 221.

[0077]

FIG. 3 is a diagram showing the construction of programs stored on a program storage medium.

[0078]

The program storage medium 300, shown in FIG. 3, representatively shows the CDROM 210 with the mechanism-control-program development support program stored thereon; the hard disk 211 with the support program installed therein; and the floppy disk 212 with the support program stored when downloaded.

[0079]

The mechanism-control-program development support program 310 is stored on the program storage medium 300 shown in FIG. 3. The mechanism-control-program development support program is constructed of a three-dimensional mechanism model simulation program 311 for operating the three-dimensional model composed of a plurality of parts including an actuator and a sensor, and a synchronizing program 312, which is added to a control program, for controlling the three-dimensional mechanism model. This synchronizing program is added to the

control program and used for executing the control program in synchronization with the three-dimensional mechanism model operated by execution of the three-dimensional mechanism model simulation program 311.

[0080]

The three-dimensional mechanism model simulation program 311 and the synchronizing program 312, which constitute the mechanism-control-program development support program 310, perform the following communication with each other when they are executed within the computer system shown in FIG. 1. With the communication, operation of the three-dimensional mechanism model is synchronized with execution of the control program. That is, the three-dimensional mechanism model simulation program 311 notifies the synchronizing program 312 of an execution time for the control program. In response to the notification, the synchronizing program 312 allows execution of the control program by the amount of the notified execution time. If the control program is executed for the execution time, the synchronizing program 312 interrupts the execution of the control program and notifies the three-dimensional mechanism model simulation program 311 that the execution of the control program for the execution time has been completed. In response to the notification from the synchronizing program 312, the three-dimensional mechanism model simulation program 311 operates the three-dimensional mechanism mode by an amount of operation that corresponds to the amount that the operating program is controlled for the

execution time. Then, three-dimensional mechanism model simulation program 311 determines the execution time and notifies the synchronizing program 312 of the determined execution time. If this is repeated, the operating program is executed in synchronization with the three-dimensional mechanism model and therefore the three-dimensional mechanism model operates according to the control for the operation program execution.

[0081]

FIG. 4 is a diagram showing an example of a computer network where an example of the mechanism-control-program development support system of the present invention is constructed.

[0082]

FIG. 4 shows two computer systems 400, 500 connected via a communication network 600. In this example, the computer system 400 constitutes of the three-dimensional mechanism model simulator of the present invention and the computer system 500 constitutes the control program processor of the present invention.

[0083]

The computer systems 400, 500 are equipped with main body portions 401, 501 having incorporated a CPU, a RAM, a hard disk, a communication modem, etc.; CRT displays 402, 502 for performing image display on display screens 402a, 502a in response to instructions from the main body portions 401, 501; keyboards 403, 503 for inputting user's instruction and character information; and mice 404, 504 for inputting instructions

corresponding to icons or the like, displayed at arbitrary positions on the display screens 402a, 502a by specifying the positions.

[0084]

The main body portions 401, 501 further have floppy disk loading slots 401a, 501a and CD-ROM loading slots 401b, 501b into which floppy disks and CD-ROMs are freely removably loaded. And inside the main body portions 401, 501, floppy disk drivers and CD-ROM drivers are also incorporated for driving the loaded floppy disks and CD-ROMs.

[0085]

In this example, the three-dimensional mechanism model simulation program 311 (see FIG. 3) in the mechanism-control-program development support program is installed in the computer system 400, and the synchronizing program 312 is installed in the computer system 500. A three-dimensional mechanism model is constructed in the computer system 400, and the three-dimensional mechanism model simulation program 311 installed in the computer system 400 operates the three-dimensional mechanism model constructed in the computer system 400. On the other hand, a control program is also stored in the computer system 500 for controlling a product when actually manufacturing the three-dimensional mechanism model constructed in the computer system 400. The synchronizing program installed in the computer system 500 is united with the control program and controls the three-dimensional mechanism that operates within the computer

system 500. The communication between the three-dimensional mechanism model simulation program 311 and the synchronizing program 312 is performed via the communication network 600.

[0086]

The mechanism-control-program development support system of the present invention also includes a case where the three-dimensional mechanism model simulator and the operating-program processor are respectively incorporated into separate single systems (in this example, the computer system 400 and the computer system 500), as shown in FIG. 4.

[0087]

In addition, the mechanism-control-program development support system of the present invention includes the following case. That is, a control program is loaded into a microcomputer chip connecting a power supply and crystal together. For example, the microcomputer chip is connected to the computer system 400 in FIG. 4 through an I/O board. An electrical signal from the microcomputer chip, which is originally transmitted to a motor driver, is input to the computer system 400 through the I/O board. In the computer system 400, the three-dimensional mechanism model is operated and the same signal as a signal that is originally output from a sensor is output. The output signal is received by the microcomputer chip.

[0088]

In this manner, there can be realized a circumstance in which the three-dimensional mechanism model is operated within

the computer system 400 and the control program is operated with the microcomputer chip in a nearly actual operating state.

[0089]

FIG. 5 is a block diagram showing the hardware of the computer system constituting the computer network shown in FIG. 4.

[0090]

In this embodiment, the computer systems 400 and 500 shown in FIG. 4 are the same in hardware constitution, so a description will be given of the computer system 400.

[0091]

In FIG. 5 there are shown a CPU 421, a RAM 422, a hard disk controller 423, a floppy disk driver 424, a CD-ROM driver 425, a mouse controller 426, a keyboard controller 427, a display controller 428. They are interconnected via a bus 420.

[0092]

FIG. 5 also shows a hard disk 411, a floppy disk 412, a CD-ROM 410, a mouse 404, a keyboard 403, and a CRT display 402. These elements are identical in operation with the corresponding elements shown in FIG. 2, so a detailed description thereof is omitted.

[0093]

In FIG. 5 a modem 429 is shown as an element not shown in FIG. 2. This modem 429 is connected to the communication network 600 shown in FIG. 4, and the communication between the two computer systems 400, 500 are performed via the modem 429.

[0094]

While two kinds of hardware constitution have schematically been shown in FIGS. 1 and 2 and FIGS. 3 and 4, the mechanism-control-program development support system constructed within the computer system or computer network will hereinafter be described in detail.

[0095]

FIG. 6 is a conceptual diagram of the mechanism-control-program development support system.

[0096]

A three-dimensional mechanism mode (hereinafter referred to as a "3D model") is constructed in the three-dimensional mechanism model simulator (hereinafter referred to as a "3D model simulator"). On the other hand, a control program for controlling the 3D model is installed within the control program processor.

[0097]

The control program in the control program processor is used for controlling a product equivalent to the 3D model when actually manufacturing the product and is coupled with a synchronizing program for synchronizing with the operation of the 3D model that operates within the 3D model simulator.

The 3D model is a three-dimensional model for the product, defined by the three-dimensional mechanism data defining the relations between the three-dimensional shapes of a plurality of links (constituting the product) and the postures of the links. The 3D model of a link is moved by changing the posture of the link.

[0098]

An actuator signal, for operating an actuator such as a motor, is transmitted from the control program processor to the 3D model simulator. In addition, a sensor signal indicating the ON/OFF state of a sensor is transmitted from the 3D model simulator to the control program processor. Furthermore, synchronous signals are transmitted and received between the control program processor and the 3D model simulator.

[0099]

FIG. 7 is a schematic diagram illustrating how the 3D model simulator processes the sensor and the motor when defining them. FIG. 8 is a schematic diagram illustrating a simulation process that is performed by the 3D model simulator.

[0100]

The 3D model is constructed by a three-dimensional CAD, etc. For the constructed 3D model, the user defines the motor and the sensor with an input device such as a keyboard, as shown in FIG 7. In performing practical simulation after the definition, the posture of a line, corresponding to an actuator instructing signal (in this embodiment, a motor-rotation instructing signal) received from the control program processor, and defined as the motor (the link defined as the motor will hereinafter be referred to as a "motor link"), is varied. Based on the mechanism data, other links connected with the motor link are moved by the amount that corresponds to the change in posture of the motor link. As a result, the posture of the link, defined as the sensor (the link defined as the sensor

will hereinafter be referred to as a "sensor link"), is also affected. Then a sensor signal corresponding to the posture of the sensor link is generated and transmitted to the control program.

[0101]

FIG. 9 is a timing diagram illustrating the operating timing, during simulation, between the 3D model simulator and the control program.

[0102]

The hatched portions in FIG. 9 each indicate a period during which the 3D model simulator or the control program operates. As shown in FIG. 9, the 3D model simulator and the control program repeat the operating and stopping cycles alternately. In each cycle, the 3D model simulator and the control program operate in the following procedure. The details of synchronization between the 3D model simulator and the control program will be described later.

[0103]

(1) The 3D model simulator performs simulation for a simulation time Δt, based on an actuator signal, and outputs a sensor signal.

[0104]

(2) The 3D model simulator determines a simulation time Δt for the next loop from the possibility of interference of the current 3D model and transfers both the determined simulation time Δt and an operation start instruction (Ready Flag) to the control program.

[0105]

(3) The control program receives the time Δt and the operation start instruction (Ready Flag) and then performs calculation (control) for the time Δt by a multi-task, whereupon the control program sends an operation termination signal (Loop Flag) to the 3D model simulator to interrupt the operation of the 3D model simulator until receiving the next operation start instruction (Ready Flag).

[0106]

(4) The 3D model simulator starts simulation having an amount corresponding to the time Δt upon receipt of the operation termination signal (Loop Flag) of the control program.

[0107]

The 3D model simulator and the control program operate alternately by repeating the foregoing procedure.

[0108]

FIG. 10 is a flow diagram illustrating the operation of the 3D model simulator at the time of simulation.

[0109]

The simulation starts with initialization (step S101) to initialize the posture of the individual links of the 3D models. Then a time interval (simulation interval) Δt during which the simulation is performed is sent to the control program side (step S102). If a signal indicating that the simulation interval Δt has been received is received from the control program processor (step S103), a calculation start instruction is sent to the control program (step S104). If the control

program processor performs calculation for the simulation interval Δt, and the termination of the calculation is notified (step S105), all the motors and sensors of the 3D model being simulated are initialized as being unprocessed with respect to the current process corresponding to a quantity of Δt (step S106).

[0110]

Then in step S201, it is judged whether or not an unprocessed motor is present, and steps S202 through S207 are repeated for each unprocessed motor. Namely, in accordance with an actuator signal for a motor to be processed, received from the control program side, the rate of rotation of the motor is determined (step S202), and the motor posture that is moved for the interval Δt is determined from the rate of rotation (step S203). Then the postures of other parts, correlated with the current motor according to the above-mentioned mechanism relation definition, are changed (step S204), and the shortest distance d between the individual parts is measured (step S205). In steps S206, S207, the shortest distance D is obtained from among all the individual shortest distances d measured when the individual motors are rotated. If the posture change during the time Δt is completed for each motor in this manner, the resultant 3D model is displayed on the screen (step S208).

[0111]

After that, it is judged whether or not an unprocessed sensor is present (step S301), and the ON/OFF state of a sensor signal corresponding to the posture of a sensor link is detected

for each unprocessed sensor (step S302). If the process is completed for all the sensors, a sensor signal related to all these sensors is output to the control program (step S303). Furthermore, a sampling interval Δt is determined according to the shortest inter-part distance D (step S401), whereupon the procedure returns to step S102 to send the determined interval Δt to the control program.

[0112]

In the 3D model simulator, the foregoing process is repeatedly executed, whereby the 3D model is simulated in synchronization with the control program.

[0113]

FIG. 11 is a flow diagram illustrating a partial flow (a portion enclosed by a dot-and-dash line in FIG. 10) that can be substituted for part of the process flow shown in FIG. 10.

[0114]

In FIG. 11, step S210 is inserted between step S202 and step S203. At this step s210, it is judged whether or not the rate of rotation of the motor being currently processed is zero ($\omega=0$). When the motor does not rotate during the current interval Δt ($\omega=0$), the procedure advances to step S211 to assume the distance to be $d=\infty$ without measuring the inter-part distance d for the operation of the motor, whereupon the procedure advances to step S206. In the judgment in step S206, the distance $d=\infty$ is excluded from the arithmetic operation for obtaining the distance D.

[0115]

A considerable amount of calculation is required to obtain the inter-part distance d. Therefore, when the motor does not rotate it is possible to reduce the amount of calculation to speed up the simulation, by not measuring the distance d for the motor.

[0116]

FIG. 12 is a schematic diagram showing how to select a link that is defined as a motor.

[0117]

In FIG. 12, the original motor is a link A. However, since the link A itself is fixed to the base, a link B, which is first driven and changed in posture on the 3D model by the original motor, is defined as a motor link for simulation. By defining the motor link in this way, it is possible to facilitate simulation.

[0118]

FIGS. 13 and 14 are schematic diagrams showing how to select a link that is defined as a sensor.

[0119]

In FIGS. 13 and 14, the original sensor is a link A, but a link B to be detected by the original sensor is defined as a sensor link. In accordance with the posture (position) of the link B, the posture of the link B in FIG. 13A or 14A is defined as the off-state of the sensor, while the posture of the link B in FIG. 13B or 14B is defined as the on-state of the sensor. Thus, in the case of the sensor, as with the case

of the motor, a link to be detected by the original sensor is defined as the sensor (sensor link), and the on-state and off-state are defined in accordance with the posture of the sensor link. Therefore, it is possible to facilitate simulation.

[0120]

FIG. 15 is a schematic diagram illustrating motor types.

[0121]

In this embodiment, two methods, i.e., "stepless speed change" and "stepwise speed change", are prepared to define a motor.

[0122]

"Stepless speed change" is the method of defining a proportional constant between a speed instruction value and a target speed. For the stepless speed change method, two more methods, i.e., a method of proportion irrespective of positive/negative, and a method of regarding the "most significant bit of a binary number as a sign," are prepared. In the former method, if a proportional constant is 100, a binary number 101 (-3 in decimal notation) is converted to -300 rpm (where the minus sign indicates rotation in the reverse direction). And in the latter method, if a binary number 101, for example, is divided into two parts, i.e. the most significant bit '1' and the remaining bits '01'. And the most significant bit '1' is assumed to be a minus sign ('0' is a plus sign), while the remaining string of bits '01' is assumed to be a numerical value (1 in decimal notation). And -1, which is the

result of the combination of the minus sign (-) and the numerical value (1), is multiplied by a proportional constant 100 to obtain a converted value -100 (rpm).

[0123]

For "stepwise speed change," a correlation table between speed instruction values and target speeds is defined. For example, in FIG. 15, -200 rpm is defined for 101 in binary notation; if "101" is input as a speed instruction value, -200 rpm is output as a target speed.

[0124]

According to the present embodiment, since the above-mentioned various kinds of defining methods are prepared, it is possible to improve a degree of freedom in defining a motor.

[0125]

FIG. 16 is a graph showing an initial change in the rotation rate ω of a motor when the motor is regarded as a primary delay system.

[0126]

As a change in the rotation rate of a motor is treated as a primary delay system of FIG. 16, simulation is performed in consideration of a delay due to the load of the motor and the influence of the delay. If the motor is treated as a primary delay system, the user can specify the characteristics of the motor only by inputting a settling time 4T, thereby minimizing a troublesome input operation.

[0127]

Assuming that the motor is treated as a primary delay system, an equation for obtaining a quantity of rotation $\Delta\theta$ of the motor in the simulation interval Δt is derived.

[0128]

The motor speed ω at time t is expressed by the following Equation (1):

[0129]

[Expression 1]

$$\omega = \omega_o + (\omega_i - \omega_o)(1 - e^{-1/T}) \quad \ldots (1)$$

[0130]

where ω_i is the target speed, ω_o is the motor speed before the target speed instruction, and T is a constant proportional to the settling time 4T indicating a characteristic of the motor. Assuming that the motor speed ω at the time t is ω_{k-1}, the following Equation (2) is obtained by the above Equation (1):

[0131]

[Expression 2]

$$\omega_{k-1} = \omega_o + (\omega_i - \omega_o)(1 - e^{-1/T}) \quad \ldots (2)$$

[0132]

Similarly, the motor speed at time $t + \Delta t$ is obtained as follows:

[0133]

[Expression 3]

$$\omega_k = \omega_o + (\omega_i - \omega_o)(1 - e^{-(t + \Delta t)/T}) \quad \ldots (3)$$

[0134]

Subtracting Equation (2) from Equation (3) yields the following Equations:

[0135]

[Expression 4]

$$\omega_k - \omega_{k-1} = (\omega_i - \omega_o)(1 - e^{-(t + \Delta t)/T}) - (\omega_i - \omega_o)(1 - e^{-1/T})$$

$$= (\omega_i - \omega_o)\{(1 - e^{-(t + \Delta t)/T}) - (1 - e^{-1/T})\}$$

$$= (\omega_i - \omega_o)(e^{-1/T} - e^{-(t + \Delta t)/T})$$

$$= (\omega_i - \omega_o)e^{-1/T}(1 - e^{-\Delta t/T}) \quad \ldots (4)$$

$$\omega_k = \omega_{k-1} + (\omega_i - \omega_o)e^{-1/T}(1 - e^{-(-\Delta t)/T}) \quad \ldots (5)$$

[0136]

Here, Equation (2) is rewritten as follows:

[0137]

[Expression 5]

$$\omega_{k-1} - \omega_o = (\omega_i - \omega_o)(1 - e^{-1/T})$$

$$= (\omega_i - \omega_o) - (\omega_i - \omega_o)e^{-1/T}$$

$$(\omega_i - \omega_o)e^{-1/T} = \omega_i - \omega_o - \omega_{k-1} + \omega_o$$

$$= \omega_i - \omega_{k-1} \quad \ldots (6)$$

[0138]

Substituting Equation (6) into Equation (5) gives the following Equation.

[0139]

[Expression 6]

$$\omega_k = \omega_{k-1} + (\omega_i - \omega_{k-1})(1 - e^{-\Delta t/T}) \quad \ldots (7)$$

[0140]

The amount that the motor rotates for Δt is obtained by integrating the Equation (7) with respect to Δt.

[0141]

[Expression 7]

$$\Delta\theta = \int_0^{\Delta t} \omega_{k-1} + (\omega_i - \omega_{k-1})(1 - e^{-1/T})dt$$
$$= \omega_{k-1}\Delta t + (\omega_i - \omega_{k-1})\{\Delta t - (1 - e^{-\Delta t/T})\} \quad \ldots (8)$$

[0142]

Assuming the motor is treated as a primary delay system, the rotation amount $\Delta\theta$ of the motor for a simulation time Δt can be obtained according to the above Equation (8) by the target speed ω_i, the simulation time Δt, the motor speed ω_{k-1} at the time before the simulation time Δt, and the settling time 4T.

[0143]

The motor does not need to be operated as expressed by the Equation (8). For instance, the speed may be fixed to ω_i after the lapse of a settling time 4T since the target speed ω_i for the motor varied.

[0144]

FIG. 17 is a flow diagram illustrating how a motor driving process in the simulator is performed with a primary delay considered.

[0145]

If the simulation interval Δt is determined based on the inter-part distance (step S501), and the actuator signal is input from the control program (step S502), the target speed ω_i is determined according to the motor definition of FIG. 15 by making reference to the proportional constant or correlation table (step S503).

[0146]

In step S504, it is judged whether or not the target speed ω_i has varied. If the current target speed ω_i has varied, the procedure advances to step S505 so that t_{total}, indicating an elapsed time from the time point when the target speed ω_i has varied, is initialized to zero. Then, for the next judgment in step S504, ω_i is stored to update ω_{iold} (step S506).

[0147]

If $\omega_i = \omega_{iold}$ as the result of judgement, the procedure goes directly to step S507. If $\omega_i \neq \omega_{iold}$, the procedure advances to steps S505 and S506 and then advances to S507. In step S507, it is judged whether or not the elapsed time t_{total} from the time point when the target speed ω_i has varied reaches the settling time 4T. If t_{total} does not reach 4T, the procedure advances to step S508 to obtain the rotation speed ω of the motor in accordance with the Equation (7), and then advances to step S509 to obtain the rotation amount $\Delta\theta$ of the motor in accordance with the Equation (8).

[0148]

If, on the other hand, $t_{total} \geqq 4T$ as the result of judgement

in step S507, the procedure advances to step S510 to set the rotation speed ω of the motor to a value equal to the target speed ω_i, and then advances to step 511 to set the rotation amount $\Delta\theta$ of the motor to an amount of rotation that the motor rotates for the simulation time Δt at the target speed ω_i.

[0149]

Even in the case where the procedure goes through steps S508, S509, or in the case where the procedure goes through steps S510, S511, t_{total} increases in step S512 by Δt and then the total rotation amount θ increases by $\Delta\theta$.

[0150]

As the process of FIG. 17 is repeatedly executed in synchronization with the control program, the motor rotates at a varying speed that is obtained when the motor is considered as the primary delay system of FIG. 16.

[0151]

FIGS. 18 through 20 are explanatory diagrams illustrating how to define a sensor using an interference check.

[0152]

FIG. 18 shows an example in which the posture of a contact sensor S1 is determined by a combination of postures of plural motors M. In this illustrative example, if the method, described above in connection with FIGS. 13 and 14, of considering as a sensor a link detected by an actual sensor is adopted, the ON/OFF state of a sensor S1 would be defined by the combination of the respective postures of plural motors M, which would require a much more complicated definition.

Consequently, an interference check is made between the sensor and the base; the sensor assumes the on or off state, depending on interference between the sensor S1 and the base. With this arrangement, it is possible to determine the ON/OFF state of the sensor S1 without requesting the user to perform any complicated definition.

[0153]

FIG. 19 shows another example in which a photoelectric sensor repeatedly assumes the on and off state as the slits are moved. In this illustrative example, a bundle of light is considered to be a single virtual link, as shown in FIG. 20. The ON/OFF state of the sensor is judged, depending on the presence of interference between the virtual link and the slit.

[0154]

In this case, for a check link for checking interference with the virtual link, the link may be limited to a link if it is beforehand known as shown in FIG. 19. Or even when if a check link is unknown, a link representing the original sensor itself is excluded since it is normally in interference (contact) with the virtual link.

[0155]

The sensor-output determining method using the virtual link should by no means be limited to the photoelectric sensor shown here. For instance, it is also applicable to a contact-type position detecting sensor in which the contact of a micro switch is defined as a virtual link.

[0156]

The sensor defining method described above in connection with FIGS. 13 and 14 is particularly effective in expressing a sensor, which detects single-degree-freedom displacement, such as a potentiometer, encoder, etc., which detect an angle. However, if the definition method is applied to a sensor in which the posture is determined as the result of a combination of multi-degree freedoms (e.g., the above-mentioned sensor (FIG. 18) located so that the output is determined by a combination of the postures of plural motors), or if the definition method is applied to a sensor which frequently assumes an ON/OFF state (e.g., the above-mentioned slit counter (FIG. 19)), the user would be requested to perform a complicated definition operation. Consequently it is preferable to adopt the illustrated method of determining the ON/OFF state of the sensor by the presence of the interference between links.

[0157]

FIG. 21 is a timing diagram illustrating signals that are transmitted and received for obtaining synchronization between the simulator and the control program. FIG. 22 is a flow diagram illustrating a synchronizing program for obtaining synchronization with the operation of the 3D model for realizing the synchronization between the simulator and the control program. FIG. 23 is a flow diagram illustrating the process that is performed by the simulator to realize the synchronization between the simulator and the control program. Here, the synchronizing program is a program that is added to the control

program for controlling the motion of a product corresponding to the 3D model when manufactured.

[0158]

A Loop Flag, a Ready Flag and a Time Scale, all of which are shown in FIG. 21, are signals respectively having the following meanings:

[0159]

(1) Loop Flag:

A signal to be transferred from the control program to the simulator, indicating whether or not the control program is currently in operation.

[0160]

(2) Ready Flag:

A signal to be transferred from the simulator to the control program, indicating that simulation corresponding to a single simulator interval Δt has been completed by the simulator.

[0161]

(3) Time Scale:

A signal to be transferred from the simulator to the control program, representing the simulation interval Δt determined based on the shortest inter-part distance by the simulator.

[0162]

In general, the control program has portions, which operate in a fixed cycle, such as a main loop, a timer interrupt routine, etc. It is preferable that the synchronizing program added to the control program for a synchronization process be a program with a minimum scale. In this example, the time scale

is specified with the number of times (n) that the control program passes through a routine which operates in this fixed cycle, not with specific time such as 5 msec, 2 sec, etc.

[0163]

The routine shown in Fig. 22 represents it, and in step S602 it is judged whether or not n is 0. If n is not 0, the procedure advances to step S603 to decrement n by 1, and shifts to the execution of the original process of the control program. Only when n=0 as the result of judgement in step S601, steps S603 through S607 for synchronization are executed.

[0164]

For explaining FIGS. 21 through 23, assume that in the simulator, the current simulation is being performed and therefore Ready Flag is in a low level. Also, assume that in step S604 of Fig. 22, the control program has waited for the Ready Flag to go high. At that time, in the routine of FIG. 23 representing the synchronization process in the simulator, the simulation in step S703 is being performed.

[0165]

Upon completion of the simulation, the simulator shifts the procedure to step S704 of FIG. 23, in which the next simulation interval (Time Scale) Δt (represented by the number of operations (n) of the control program, as described above) is determined. And in step S705, the Time Scale Δt (n) is transmitted to the control program. The procedure further advances to step S706, in which the Ready Flag is changed to a high level.

[0166]

At the control program side, the procedure escapes from step S604 of FIG. 22 and advances to step S605, in which Time Scale n is read. The procedure further advances to step S606, in which the Loop Flag is changed to a high level. Subsequently, in step S607, the procedure advances to the original process after the Ready Flag being sent from the simulator goes to low. As described previously, the control program are repeatedly executed in fixed cycles, and in step S601, it is judged whether or not $n=0$, and if $n \neq 0$, the procedure advances to step S602, in which n is decremented by 1. This is repeated. If $n=0$ as the result of judgement in step S601, the procedure advances to step S603, in which the Loop Flag is changed to a low level. Subsequently, in step S604, the procedure waits for the Ready Flag, transmitted from the simulator, to go to a high level.

[0167]

On the other hand, at the simulator side, in step S706 of FIG. 23 the Ready Flag is changed to a high level. In step S707, the procedure waits for the Loop Flag, transmitted from the control program side, to go to a high level. If the Loop Flag goes to a high level, the procedure advances to step S708 to change the Ready Flag to a low level, and then advances to step S702 to wait for the Loop flag to go to a low level. If the Loop Flag goes low, the procedure advances to step S703, in which simulation corresponding to the simulation interval Δt is performed.

[0168]

The foregoing process is repeated, and in the simulator and the control program, the respective steps are executed while mutual synchronization is being established between the simulator and the control program.

[0169]

Now, a description will be given of a method of evaluating the possibility of interference between links (parts).

[0170]

FIG. 24 is an explanatory diagram illustrating how to evaluate the possibility of interference. In this example, a link A is moving toward a link B at velocity v.

[0171]

FIG. 24A shows two spaced links (the link A and the link B) which have the possibility of interfere. In this case, it is judged that the possibility of interference is low. And, a relatively large value is set as the simulation interval Δt. The link A moves $v\Delta t$ with single simulation. In this case, simulation can be performed at high speeds.

[0172]

FIG. 24B shows the link A and the link B being spaced a small distance from each other. In this case, it is judged that the possibility of interference is high. And a relatively small value is set as the simulation interval Δt. In this case, the distance $v\Delta t$ that the link A moves with single simulation is small because Δt is small. Therefore, high-precision simulation can be performed.

[0173]

Thus, in the present embodiment, the simulation interval Δt is changed in accordance with the distance between the individual parts, whereby the compatibility and harmony between high-precision simulation and high-speed simulation can be achieved.

FIG. 25 is a graph showing various kinds of relationships between the inter-part distance d and the simulation interval Δt.

[0174]

As shown in these examples, the simulation interval Δt is determined with respect to the inter-part distance d in such a manner that the simulation will not be performed at an extremely low speed even if the distance d is short and at an extremely high speed even if the distance d is long.

[0175]

Now, a description will be made of a method of dividing parts into groups for evaluating the possibility of interference.

[0176]

FIG. 26 is an explanatory diagram of grouping of parts.

[0177]

In FIG. 26, there are shown 5 links, l_0 through l_5, and consider the case where the link l_3, defined as a motor link, is moved. In this case, the links that are influenced in posture by movement of the motor link l_3 are links l_3 through l_5, which can be regarded as a single rigid body with respect to the movement of the motor link l_3. Similarly, the links l_0 through l_2, which

are not influenced in posture by movement of the motor link l_3, can be regarded as a single rigid body. Hence, in the case of moving the motor link l_3, the remaining links are divided into two groups, one group of links whose postures are influenced by moving the motor link l_3 and the other group of links whose postures are free of such influence. Each of the two groups is treated as a single rigid body, whereupon the shortest distance between these two rigid bodies is obtained. The shortest distance can be considered to be one of the shortest distance candidates to determine the next simulation interval Δt. However, the link l_3 and the link l_2 (e.g., a motor shaft and a gear) in FIG. 26 are separately incorporated in different groups, though they are normally in contact with each other. Consequently, the above-mentioned two rigid bodies will be considered as interfering with each other. Hence, in the present embodiment, links are divided into two groups, as described above. Then, the distance between the parts is obtained. Among the links in engagement or interference with one another, the link (i.e., the link l_2 in the example of FIG. 26), belonging to one group whose individual links do not move when a motor link of interest is moved, is excluded from that group, whereupon the two groups are individually treated as rigid bodies and then a distance between these two rigid bodies is obtained.

[0178]

FIG. 27 is an explanatory diagram illustrating grouping in the case where a plurality of motor links are defined in

a single three-dimensional model.

[0179]

FIG. 27A shows a 3D model in which three motor links of motors M1 through M3 are defined. The grouping, described with reference to FIG. 26, is performed when only the motor 1 is moved (FIG. 27B), when only the motor 2 is moved (FIG. 27C), and when only the motor 3 is moved (FIG. 27D). During execution of the simulation, if an object motor link is moved, the shortest distance between the groups of the motor links is obtained. Among the plurality of 'shortest distances' obtained when the plural motor links are moved, the shortest distance is further obtained (see steps S201 through S207 in FIG. 10).

[0180]

FIG. 28 is an explanatory diagram illustrating regrouping of the grouped parts.

[0181]

According to the method described above in connection with FIG. 27, for each motor link, almost all of the parts constituting the 3D model would belong to either group (rigid body). As a result, the amount of data to define the individual rigid body would be huge, which would often require a very large memory capacity. Consequently, as shown in FIG. 28, among a set of parts belonging to the group (the group (D), (E) or (F) in FIG. 28) in which the individual parts do not move when each motor link is moved, a common portion is extracted as a single rigid body, and each group (D), (E) or (F), excluding the common portion, is treated as a single rigid body. In measuring

distance, a rigid body which consists of the common portion may be combined with a rigid body excluding the common portion, and the distance between the combined rigid body and the other group (i.e., the group (A), (B) or (C) corresponding to the group (D), (E) or (F)) may be obtained. If done in this manner, the capacity of memory for storing data to define these groups can be reduced, compared with the case where each group includes a common portion (the case of FIG. 27).

FIG. 29 is a flow diagram illustrating the routine of dividing a plurality of links into a group of links which are affected in posture by movement of the motor link and a group of links which are not affected in posture by the movement of the motor link. FIG. 30 is a flow diagram illustrating a retrieval routine which is a subroutine of the routine of FIG. 29.

[0182]

In this example, links whose postures are influenced by movement of a motor link are listed in a list A, while the remaining links are listed in a list B. These lists A and B are cleared before executing the process described here.

[0183]

In this example, a motor link is first specified with a pointer P (step S801 in Fig. 29), and then the specified motor link is added to the list A.

[0184]

Next, it is judged whether or not there is present a child link in the link specified with the pointer P. When the pointer

P specifies the motor link l_3 in the link mechanism of FIG. 26, the term "child link" means one (link l_4 here) of the links which moves together with the motor link l_3 in response to movement of the motor link l_3. Even if the link l_4 is a motor link, the current motor link of interest is the link l_3 and hence the link l_4 becomes a child link of the link l_3. Even though a plurality of identical child links are specified with the pointer P, the child link is limited to only one of the identical child links and the remaining child links are defined as "brother links" of the single child link.

[0185]

If the link specified with the pointer P has a child link, the procedure advances to step S804, in which the pointer P is moved so as to specify the child link, and a retrieval routine of FIG. 30 is called. The retrieval routine will be described later. Upon completion of execution of the retrieval routine in step S805, the procedure advances to step S806. In step S803, if it is judged that the link specified with the pointer P has no child link, the procedure advances directly to step S806.

[0186]

In step S806, it is judged whether or not the link specified with the pointer P is a driving link for driving a passive part, such as a gear, a cam, etc.

[0187]

If the link specified with the pointer P is a driving link for driving a passive part, the procedure advances to step

S807, in which the pointer P is moved so as to specify a passive part that is to be driven by the link specified with the pointer P, and the retrieval routine of FIG. 30 is called in step S808. Upon completion of execution of the retrieval routine in step S808, the procedure advances to step 809. In step S806, if it is judged that the link specified with the pointer P is not a link for driving a passive part, the procedure advances directly to step S809.

[0188]

At the stage where the procedure has advanced to step S809, the list A has been completed, and in step S809, all the links, excluding the links listed in the list A, are stored in the list B. Next, an interference check is made between all the links listed in the list A and all the links listed in the list B (step S810). Among a set of links interfering each other, the link listed in the list B is removed from the list B.

[0189]

In the retrieval routine of FIG. 30, in step S901 it is judged whether or not the link specified with the pointer P has already been listed in the list A. If it has already been listed in the list A, the procedure returns directly to the original routine from which the retrieval routine was called. In step S901, if it is judged that the link specified with the pointer P has not been listed in the list A, the procedure advances to step S902, in which the link specified with the pointer P is added to the list A. In step S903, it is judged whether

or not the link specified with the pointer P has a child link. If the link specified with the pointer P has a child link, the procedure advances to step S904, in which the pointer P is moved so as to specify the child link. In step S905, the retrieval routine of FIG. 30 is called again. In this manner, this retrieval routine is often repeatedly called. Upon completion of execution of the retrieval routine, the procedure returns to the step in which the retrieval routine has been called (e.g., when called in step S905, the procedure returns to step S905).

[0190]

In step S906, it is judged whether or not the link specified with the pointer P has a brother link. If the specified link has a brother link, the procedure advances to step S907 where the pointer P is moved so as to specify the brother link, and then advances to step S908 where the retrieval routine is called again.

[0191]

In step S909, it is judged whether or not the link specified with the pointer P is a driving link for driving a passive part such as a gear, a cam, etc. If the specified link is a driving link, the procedure advances to step S910 where the pointer P is moved so as to specify the driving link, and then advances to step S911 where the retrieval routine is called again.

[0192]

FIG. 31 is a diagram showing a link mechanism model for explaining operation of the routines of FIGS. 29 and 30. FIG. 32 is a diagram illustrating a data structure for the link

mechanism model shown in FIG. 31.

[0193]

As shown in FIG. 31, the link A is a motor; the links B and C are a shift and a gear, respectively, which rotate as the link A (motor) rotates; and the link D is a gear meshing with the link C. And the link E is a shaft which rotates along with the link D (gear) as the last-named link rotates, and the link F is a gear meshing with the link D (gear).

[0194]

In the data structure shown in FIG. 32, the link A, the link D and the link F are arranged in parallel; the link B is the child link of the link A; the link C is a brother link of the link B; and the link E is the child link of the link D.

[0195]

For the link mechanism model of FIGS. 31 and 32, consider that the list A is generated by executing the routine of FIGS. 29 and 30. In this example, the retrieval routine starts with the link A (motor). Namely, the pointer P is moved so that the link A is specified in step S801 of FIG. 29. In the following explanation of the retrieval routine, each capital alphabetical letter in a parenthesis represents a link listed in the list A.

[0196]

(1) With pointer P = link A, the retrieval routine starts. (A)

(2) The pointer P is moved from the link A to the child link B, and the retrieval routine is executed. (A, B)

(3) The pointer P is moved from the link B to the brother link C, and the retrieval routine is executed. (A, B, C)

(4) The pointer P is moved to the link D, which is a passive part of the link C, and the retrieval routine is executed. (A, B, C, D)

(5) The pointer P is moved from the link D to the child link E, and the retrieval routine is executed. (A, B, C, D, E)

(6) The pointer P returns to the link D by "RETURN."

[0197]

(7) The pointer P is moved from the link D to the link F, which is a passive part of the link D, and the retrieval routine is executed. (A, B, C, D, E, F)

(8) The pointer P returns to the link D by "RETURN."

(9) The pointer P returns to the link C by "RETURN."

(10) The pointer P returns to the link B by "RETURN."

(11) The pointer P returns to the link A by "RETURN."

(12) The retrieval routine ends.

With the foregoing procedure, all the links (links A through F) shown in FIG. 31 are listed in the list A.

[0198]

FIG. 33 is an explanatory diagram illustrating a method of evaluating the possibility of interference in the group of links whose postures are changed by movement of a single motor link.

[0199]

The interference due to the movement of the motor link,

as described above in connection with FIG. 26, can occur not only between the group of links whose postures are influenced by the movement of the motor link and the group of links whose postures are not influenced by the movement of the motor link, but also within the group of links whose postures are influenced as shown in FIG. 31. Hence, for finding a set of links which can interfere with each other in the group, an interference check is made between all the individual links in the group, whose postures are influenced by the movement of the motor link, before start of simulation, while the motor link is moved little by little. By making an interference check of all possible link sets, the set of interfering links set is added to the object links where a distance between links is calculated when the actual simulation is performed. The set of interfering links are also used as one of the shortest-distance candidates for determining the simulation interval Δt is determined. In this manner, accurate simulation can be performed. However, in the group in which the postures of the individual links are influenced by the movement of the motor link, a set of links normally in contact with each other are present like the motor and gear shown in FIG. 31. Hence, a check of interference is made by moving the motor link little by little, and a set of links, which always interfere and contact with each other at all the postures of the motor link (e.g., a set of companion links interconnected by a gear, etc.), are excluded from the object links between which distance is calculated.

[0200]

FIG. 34 is an explanatory diagram illustrating a method of extracting, from the links affected in posture by movement of the motor link, a set of links between which distance is calculated during simulation.

[0201]

FIG. 34A shows all parts constituting a group in which the postures of the individual parts vary due to movement of a single motor. In an illustrative example shown in FIG. 34B, the motor is turned intermittently at pitches of 36 degrees from -180 to 180 degrees, and an interference check is made for each 36-degree movement. In this case, the turning of the motor and the interference check are performed eleven times. As a result, data is obtained for a set of interfering links and the frequency of interference, as shown in FIG. 34C. The interference check was made eleven times, and the interference occurred eleven times between the links B and E and between the links D and E. Therefore, these two sets are excluded from the object sets whose inter-link distances are to be measured during simulation.

[0202]

Now, a description will be given of a second embodiment of the 3D mechanism model simulator which constitutes the mechanism-control-program support system of the present invention.

[0203]

FIG. 35 is a flow diagram illustrating the entire simulation process of this embodiment.

[0204]

A synchronization process between the simulator and the control program (step S1001), a motor process (step S1002), a joint malfunction process (step S1003), a sensor process (step S1004), and a process of determining an execution time ΔT during which the control program is performed (step S1005), are repeatedly executed.

[0205]

The synchronization process between the simulator and the control program (step S1001) is the same as the aforementioned embodiment (see FIGS. 9 and 21), so a description thereof is omitted. The remaining processes will be described in sequence.

[0206]

FIG. 36 is a flow diagram illustrating a motor processing routine. In the flow diagram of FIG. 36, the concept of a motor failure is basically incorporated in the motor process of the first embodiment that is constructed of the portion enclosed by a dash-and-dot line in FIG. 10.

[0207]

In step S1101, ∞ is first input to the shortest distance D to calculate the shortest distance D. In step S1102 it is judged whether or not there is an unprocessed motor, and steps S1103 through S1110 are performed for each unprocessed motor.

[0208]

That is, for each unprocessed motor, it is first judged whether or not the motor has failed (step S1003). The presence

of a motor failure is previously set and stored as a flag by the user. No process is performed for a motor failure. When the motor has not failed, a posture change amount $\Delta\theta$ for the motor is determined in accordance with an actuator signal, transmitted from the control program, for the motor being currently processed (step S1004). If $\Delta\theta$ is not 0 (step S1105), the current change amount $\Delta\theta$ is added to the preceding posture θ of the motor (step S1106). A link posture movement subroutine (to be described later with reference to FIG. 37) is executed for calculating a posture change amount that a link moves when the motor link is moved (step S1107). Subsequently, the inter-part shortest distance d is obtained (step S1108), and it is judged whether or not d<D. When d<D, that is, when the currently obtained shortest distance d is shorter than the previously obtained shortest distance D (step S1109), the currently obtained shortest distance d is substituted for the shortest distance D to update the last-named distance (step S1110).

[0209]

For all the motors, upon completion of posture movement during the current arithmetic time ΔT, this motor process is terminated.

[0210]

FIG. 37 is a flow diagram illustrating a link-posture movement subroutine which is executed in step S1107 of the motor process routine shown in FIG. 36.

[0211]

First, in response to link posture movement (when called in step S1107 of FIG. 37, the posture movement of a motor link) (step S1201), it is judged whether or not there is present an uncalculated passive link in the passive links corresponding to the link whose posture was moved (step S1202). If an uncalculated passive link is present, it is judged whether or not a clutch is present between the link whose posture has currently been moved and the passive link which is passive with respect to the posture-moved link (step S1203). If a clutch is present, it is judged whether or not the posture of the clutch (part) is in an ON-position (step S1204). The presence of the clutch, and the relationship between the posture of the clutch and the ON/OFF State of the clutch, are previously defined by the user.

[0212]

If no clutch is present, or if it is present and in the ON-position, the procedure advances to step S1205 where the amount of movement of a passive part is calculated based on the relationship between a gear and a cam. In step S1206, when the passive part for which the current movement amount is calculated is regarded as a positive part, the link posture movement subroutine is called again for obtaining the amount of movement of a passive part with respect to the positive part.

[0213]

In this manner, the link posture movement subroutine, as with the retrieval routine of FIG. 30, is recursively executed. With this recursive execution, the amounts of movement of other

parts can be obtained at an increased rate, when a single motor is moved.

[0214]

FIG. 38 is a flow diagram illustrating a joint malfunction processing routine to be executed in step S1003 of FIG. 35.

[0215]

First, a three-dimensional mechanism model is displayed on the screen (step S1301). Then, it is judged whether or not the user has specified a fault link (step S1302). Next, it is judged whether or not there is present an unprocessed fault link specified by the user (step S1305). Thereafter, it is judged whether or not there is present an unprocessed interfering link (step S1310), and it is judged whether or not there is present a fault link (step S1314).

[0216]

In step S1302, if it is judged that the user has newly specified a fault link, the procedure advances to step S1303. In step S1303, a motor for driving the specified fault link is retrieved, and the motor and the moving direction of the motor are added to the list (step S1304).

[0217]

If it is judged in step S1305 that the unprocessed fault link, specified by the user, is present, the procedure advances to step S1306 where it is judged whether or not the moving direction of the motor for driving the fault link has been reversed. If it has not been reversed, the procedure advances to step S1307 where the motor is returned to its original posture.

To return the posture of the link to the original posture in accordance with the returning of the motor posture to the original posture, the link posture movement subroutine of FIG. 37 is called (step S1308). On the other hand, in step S1306, if it is judged that the moving direction of the motor has been reversed, the procedure advances to step S1309 where the motor is deleted from the fault link list.

[0218]

In this manner, an operation is simulated in which, when a fault link is moved in one direction, it is caught for some reason and is no longer movable, and if the fault link is moved in the opposite direction, the catch is released and therefore the link returns to its normal condition.

[0219]

In step S1310, if it is judged that there is present an unprocessed interfering link, the procedure advances to step S1311. In step S1311, a motor for driving the fault link is retrieved, and the original posture of the motor is restored (step S1312). To restore the original postures of other links to be driven by the motor, the link mechanism subroutine of FIG. 37 is called (step S1313).

[0220]

In this manner, it is possible to simulate a situation in which a particular link interfere with another link and cannot be moved any further.

[0221]

In step S1314, it is judged that a fault link is present,

the three-dimensional mechanism model is displayed again on the screen (step S1315). In this manner, if the simulation is viewed with the three-dimensional mechanism model displayed on the screen in step S1301, the motor (which has the fault link as a passive part) and the movements of all links (which are driven by the motor) are displayed and therefore the caught condition is expressed.

[0222]

In the illustrated example, a joint fault is realized by restoring the original posture of the motor. Alternatively, the joint fault may be realized by temporarily setting the upper limit or lower limit of the joint value to a current value.

[0223]

FIG. 39 is a flow diagram illustrating a sensor processing routine to be executed in step S1004 of FIG. 35. In the aforementioned embodiment, the sensor is limited to an ON/OFF sensor which outputs an ON-signal and an OFF-signal. However, this example, in addition to the ON/OFF sensor, includes a potentiometer, an encoder, and a defective sensor.

[0224]

In step S1401, the presence of an unprocessed sensor is judged. If an unprocessed sensor is present, the procedure advances to step S1402 where it is judged whether or not the unprocessed sensor is an ON/OFF sensor. If it is an ON/OFF sensor, an ON/OFF-sensor process is performed in step S1403. As a result of the ON/OFF-sensor process, a sensor value (ON or OFF) is output.

[0225]

In step S1402, if it is judged that the unprocessed sensor is not an ON/OFF sensor, the procedure advances to step S1404 where it is judged whether the sensor is a potentiometer or an encoder. If it is a potentiometer, the procedure advances to step S1405 where a potentiometer preprocess is performed. Then, if it is judged that the sensor is an encoder, the procedure advances directly to the encoder process in step S1406, As a result, a sensor value is output.

[0226]

FIG. 40 is a flow diagram illustrating an ON/OFF-sensor processing routine to be executed in step S1403 of FIG. 39.

[0227]

In this ON/OFF sensor process routine, an ON/OFF-sensor failure is also simulated. In step S1501, it is judged whether or not the-ON/OFF sensor in the ON state has failed. In step S1503, it is judged whether the ON/OFF sensor in the OFF state has failed. If the ON/OFF sensor in the ON state has failed, an ON-value is set as an estimated output value (step S1502). If the ON/OFF sensor in the OFF state has failed, an OFF-value is set as an estimated output value (step S1504), and the procedure advances to step S1508 where the estimated output value is output as a sensor value. Whether the failed ON-OFF sensor is in the ON state or OFF state is previously set by the user.

[0228]

If the ON/OFF sensor has not failed, the procedure advances

to step S1505 where the estimated output value (ON-value or OFF-value) is determined from the posture of the sensor link (step S1505). Subsequently, an estimated-output-value change process due to performance degradation (step S1506), and an estimated-output-value change process by chattering (step S1507), are executed.

[0229]

FIG. 41 is a flow diagram illustrating a sensor performance degradation processing routine which is executed in step S1506 of the ON/OFF-sensor process routine of FIG. 40. In this routine, the following situation is simulated. The response performance of the ON/OFF sensor is degraded and the output of the ON/OFF sensor goes to an ON-state or an OFF-state. The previous state is kept for a while and the output of the ON/OFF sensor goes to the ON-state or OFF-state with a delay.

[0230]

In step S1601, it is judged whether or not an estimated output value is equal to an estimated original output value d. If change has occurred, the procedure advances to step S1602 where it is judged whether or not the estimated output value is an ON value. If the estimated output value is an ON value, the procedure advances to step S1603 where a delay time when the sensor output changes from an OFF value to an ON value is set to Td. On the other hand, if the estimated output value is an OFF value, the procedure advances to step S1604 where a delay time when the sensor output changes from the OFF value to the ON value is set to Td.

[0231]

In step S1605, it is judged whether or not Td>0. If Td>0, the procedure advances to step S1606 where an execution time ΔT for the control program is subtracted from Td. In step S1607, the estimated output value is inverted (i.e., the estimated output value is set to the OFF value when it is the ON value and to the ON value when it is the OFF value).

[0232]

In step S1608, the current estimated output value is stored to update the original estimated value d.

[0233]

As shown in FIG. 35, the sensor process of S1004 is repeatedly executed, and the sensor performance degradation process of FIG. 41 is also executed repeatedly. When Td>0, ΔT is subtracted from Td each time this process is repeated, and the sensor out will not vary until Td≦0.

[0234]

FIG. 42 is a flow diagram illustrating an estimated output value change processing routine for chattering, which is to be executed by step S1507 of the ON/OFF-sensor processing routine of FIG. 40. In the estimated value change processing routine, chattering is simulated when the ON/OFF sensor changes from the ON state to the OFF state, or vice versa.

[0235]

In step S1701, it is judged whether or not the estimated output value is equal to the original estimated output value c. If change has occurred, the procedure advances to step S1702

where it is judged whether the estimated output value is the ON value or OFF value. If the estimated output value is the ON value, the procedure advances to step S1703 where a chattering time when the sensor output value changes from the OFF value to the ON value is set to Tc. If the estimated output value is the OFF value, the procedure advances to step S1704 where a chattering time when the sensor output changes from the ON value to the OFF value is set to Tc.

[0236]

In step S1705, the estimated output value is set to update the original estimated output value c.

[0237]

In step S1706, it is judged whether or not Tc>0. If Tc>0, the procedure advances to step S1707 where the execution time ΔT for the control program is subtracted from Tc. In step S1708, a random number representing one of 0 to 99 is generated, and the generated random number is compared with a preset possibility of occurrence (a value representing one of 0 to 99). As the result, if random number > possibility of occurrence, the estimated output value (ON or OFF) is inverted (step S1709).

[0238]

In this manner, such chattering that changes from an ON state to an OFF state at random is simulated during Tc>0.

[0239]

Now, a description will be made of the encoder process that is executed in step S1406 of FIG. 39.

[0240]

FIG. 43 is a graph showing one example of an output waveform for movement (e.g., angle) of an encoder, and FIG. 44 is a diagram showing various output waveforms of the encoder.

[0241]

As shown in FIG. 43, assume that the output waveform of the encoder is a periodic function $f(\theta)$ having a wavelength L. The function $f(\theta)$ can have selectively various kinds of waveforms, as shown in FIG. 44. The wavelength L corresponds to a single cycle of the function $f(\theta)$. For example, assuming that this encoder is a rotary encoder and outputs a single-period signal for every turn of 1.0 degree, L=1.0 (degrees).

[0242]

With the change θ converted to an electrical angle $\phi = 2\pi\theta/L$ and the function $f(\theta)$ expressed as a function $g(\phi)$, the principle of the encoder function will hereinafter be described.

[0243]

Assuming that the electrical angle of the encoder during the previous simulation is ϕ, the electrical angle at the time advanced from the previous simulation by the arithmetic time ΔT of the control program is ϕ, and that the electrical angular velocity of the encoder is ω, the amount of movement of the encoder in a simulation cycle from the previous simulation to the current simulation is $\omega\Delta T$, thus establishing

$$\phi = \phi_0 + \omega\Delta T.$$

[0244]

However, in measuring the number of pulses, the encoder

has to output $g(\phi)$ after outputting a predetermined number of pulses, because it cannot merely output $g(\phi)$ if $\omega \Delta T$ in the current simulation cycle is greater than 2π.

[0245]

FIG. 45 is a flow diagram illustrating an output processing routine for the encoder.

[0246]

In the encoder output processing routine, the encoder output $g(\phi_0)$ at the time of the termination of the previous simulation is first output (step S1801). Then it is judged whether or not a difference $\phi - \phi_0$ between the electrical angle ϕ at the time of the termination of the current simulation and the previous electrical angle ϕ_0 is $\phi - \phi_0 > \pi/2$ (step S1802), or it is judged whether or not it is $\phi_0 - \phi > \pi/2$ (step S1804). If it is $\phi_0 - \phi > \pi/2$, the procedure advances to step S1803 where $\pi/2$ is added to ϕ_0, and then advances to step S1806 where the procedure stands by for 50 μ sec. The procedure returns to step S1801 where $g(\phi_0)$ for new ϕ_0 is output. If $\phi_0 - \phi > \pi/2$, the procedure advances to step S1803 where $\pi/2$ is subtracted from ϕ_0, and then advances to step S1806 where the procedure stands by for 50 μ sec. Similarly, the procedure returns to step S1801 where $g(\phi_0)$ for new ϕ_0 is output.

[0247]

Namely, if $|\phi - \phi_0| > \pi/2$, a dispersing periodic signal, obtained when ϕ varies at pitches of $\pi/2$, is output until $|\phi - \phi_0| \leqq \pi/2$, with the upper limit set to a frequency of 5 kHz (step S1806).

[0248]

If $|\phi - \phi_0| \leqq \pi/2$, the procedure advances to step S1807 where $g(\phi)$, which is the final value of the encoder in the previous simulation, is output so that ϕ of the current simulation is updated as current ϕ_0 for the next simulation (step S1808).

[0249]

It is assumed that this encoder signal is received by a counter circuit which operates independently of the control program so that no problem would occur even when the control program is stopped during transmission of the encoder signal.

[0250]

The flow diagram of FIG. 45 expresses an encoder of the type of outputting only a single pulse signal $g(\phi)$. This diagram can also express an encoder of the type where a two-phase signal with an A phase and a B phase is output, by outputting $g(\phi + \pi/2)$ in addition to $g(\phi)$.

[0251]

As another alternative, a positive/negative signal of $\phi - \phi_0$ may be output in terms of ON/OFF so that an encoder of the type of outputting an up/down signal can be expressed.

[0252]

For preparation of the above-mentioned simulation, an offset signal, indicating a zero position at which the waveform, resolution, or electrical angle ϕ of a pulse signal (see Fig. 44) becomes zero (0), is input so that the user can define the characteristics of the encoder.

[0253]

FIG. 46 is a flow diagram illustrating an encoder processing routine which is to be executed in step S1406 of the process routine of FIG. 39.

[0254]

In this routine, an encoder failure is also considered. For simulating a three-dimensional mechanism model when the encoder fails, the following fault flags 0 through 6 are previously set:

[0255]

Fault flag 0: normal

Fault flag 1: A-phase failure at V_{max}

Fault flag 2: A-phase failure at V_{min}

Fault flag 3: B-phase failure at V_{max}

Fault flag 4: B-phase failure at V_{min}

Fault flag 5: up/down-signal failure (fixed to a high level)

Fault flag 6: up/down-signal failure (fixed to a low level)

In step S1901 of Fig. 46, change θ is converted into an electrical angle ϕ, and then it is judged whether or not the fault flag is 1 (step S1902), and it is judged whether or not the fault flag is 2 (step S1903). If the fault flag is neither 1 nor 2, the procedure advances to step S1904 where the g (ϕ) output process routine of FIG. 45 is executed. On the other hand, if it is judged in step S1902 that the fault flag is 1, the procedure advances to step S1905 where g(π/4), which corresponds to the phase failure at V_{max}, is output. On the

other hand, in step S1903, if it is judged that the fault flag is 2, the procedure advances to step S1906 where $g(3\pi/4)$, which corresponds to the phase failure at V_{min}, is output.

[0256]

In step S1907, it is judged whether or not the encoder is a two-phase output type encoder, and in step S1908, it is judged whether or not the encoder is an encoder of the type of outputting an up/down signal. If the encoder is not the two types and outputs $g(\phi)$ only, this routine is skipped.

[0257]

In step S1907, if it is judged that the encoder is the two-phase output type encoder, the procedure advances to step S1909 where it is judged whether or not the fault flag is 3, and then advances to step S1910 where it is judged whether or not the fault flag is 4. If the fault flag is neither 3 nor 4, the procedure advances to step S 1911 where the output processing routine substituting $\phi+\pi/2$ for ϕ in FIG. 45 is executed. In step S1909, if it is judged that the fault flag is 3, the procedure advances to step S1912 where $g(\pi/4)$, which corresponds to B-phase failure at V_{max}, is output. Likewise, in step S1910, if it is judged that the fault flag is 4, the procedure advances to step S1913 where $g(3\pi/4)$, which corresponds to B-phase failure at V_{min}, is output.

[0258]

Furthermore, in step S1908, if it is judged that the encoder is an encoder of the type of outputting an U/D (up/down) signal, the procedure advances to step S1914 where it is judged whether

or not the fault flag is 5, and then advances to step S1915 where it is judged whether or not the fault flag is 6. If the fault flag is neither 5 nor 6, the procedure advances to step S1916 where the moving direction of the encoder is judged. If it is upward, the procedure advances to step S1917 where a high level is output. If the moving direction is downward, the procedure advances to step S1918 where a low level is output. On the other hand, in step S1914, if it is judged that the fault flag is 5, the procedure advances to step S1917 where a high level is output regardless of the moving direction of the encoder. Likewise, in step S1915, if it is judged that the fault flag is 6, the procedure advances to step S1918 where a low level is output regardless of the moving direction of the encoder.

[0259]

Thus, not only the encoder type but the encoder failure can be expressed. In the case of an encoder, unlike the case of an ON/OFF sensor, fault flags are judged and processed in outputting a sensor signal, instead of indicating "occurrence of failure" immediately after start of processing.

[0260]

FIG. 47 is a diagram illustrating function values for a potentiometer.

[0261]

In this example, the potentiometer is realized as an encoder, which is extremely low in resolution, for outputting a triangular wave of FIG. 44D. For example, assuming that the wavelength L of FIG. 43 is set to 360 degree, it is possible

to express a potentiometer in which the output voltage V varies at an angle of $\theta_{max}-\theta_{min}=180$ degrees from the minimum value V_{min} from the maximum value V_{max}.

[0262]

FIG. 48 is a flow diagram illustrating a potentiometer preprocessing routine to be executed in step S1405 of the sensor process routine shown in FIG. 39.

[0263]

In the case of a potentiometer, there are cases where the output voltage V is fixed at the minimum value for θ equal to or less than θ_{min} and is fixed at the maximum value for θ equal to or greater than θ_{max}, as indicated by a solid line in FIG. 47. In Fig. 48, such a potentiometer is realized.

[0264]

That is, in step S2001, it is judged whether or not $\theta > \theta_{max}$ is satisfied. If it is satisfied, the procedure advances to step S2002 where θ is fixed to θ_{max}.

[0265]

Likewise, in step S2003, it is judged whether or not $\theta < \theta_{min}$ is satisfied. If it is satisfied, the procedure advances to step S2004 where θ is fixed to θ_{min}.

[0266]

In the case of the potentiometer, the encoder process of FIG. 46 is executed after the aforementioned preprocess has been performed (see FIG. 39).

[0267]

In this type of potentiometer, if the user inputs θ_{max},

θ_{min}, V_{max}, and V_{min}, the support system automatically converts each of these input values into the wavelength, amplitude and offset of a triangular wave $f(\theta)$.

[0268]

A potentiometer, whose output periodically varies, such as a stopper-less potentiometer, is treated as an encoder which outputs a triangular or sawteeth wave having low resolution.

[0269]

FIG. 49 is a flow diagram illustrating a ΔT determination processing routine to be executed in step S1005 of FIG. 35.

[0270]

In step S2101, it is judged whether or not there is present a sensor in which performance degradation (response delay or chattering) has occurred. If no such a sensor is present, ΔT is determined according to the shortest inter-part distance D obtained by the process of FIG. 36 (step S2102). On the other hand, if such a sensor is present, a possible minimum value is set for ΔT.

[0271]

The reason why ΔT for a performance-degraded sensor is set to a minimum value is for performing accurate simulation by increasing the precision of expression for performance degradation. In the case of a normal sensor, the sensor operates without any problem, since a delay time is previously set to zero.

[0272]

FIG. 50 is an explanatory diagram illustrating a method,

based on an interference check, of retrieving a joint movable range.

[0273]

The user first selects which direction has a limited value and also selects which parts are restricted by contact. Thereafter, a joint movable range is retrieved.

[0274]

In retrieving this joint movable range, the joint posture x is incremented from the initial value x_0 by Δx, and an interference check is made for each increment. As shown in FIG. 50, it is found that interference does not occur when $x=x_0+n\Delta x$, but occurs when $x_0+(n+1)\Delta x$. Subsequently, by using an amount of movement $\Delta x'$ smaller than Δx (e.g., $\Delta x'=\Delta x/10$), x is likewise incremented from $x_0+\Delta x$ by $\Delta x'$. As a result, it is also found that interference does not occur when $x=x_0+n\Delta x+n'\Delta x'$, but occurs when $x_0+n\Delta x+(n'+1)\Delta x'$.

[0275]

By repeating the foregoing process several times, it is possible to obtain a limited value for x, $x_0+n\Delta x+n'\Delta x'+n''\Delta x''\ldots$, at high speed with desired precision. However, it is considered that two parts specified by the user will not interfere with each other even when the joints are moved in any direction. Therefore, it is preferable to prevent an infinite loop by setting an upper limit for the number of retrieval operations (n) when performing the first interference position retrieval.

[0276]

FIG. 51 is an explanatory diagram illustrating how a cam relation is retrieved by an interference check, and FIG. 52 is a flow diagram illustrating a cam relation retrieving routine based on an interference check.

[0277]

In this example, the user previously specifies a driving part, a passive part, an initial posture of the driving part, an initial posture of the passive part, and a final posture of the driving part.

[0278]

In the routine of FIG. 52, the initial posture of the driving part and the initial posture of the passive part are first set to ϕ and ϕ, respectively (step S2201). Then, a non-contact posture of the passive part is retrieved and is stored as ϕ (step S2202).

[0279]

In step S2203, a set of the posture ϕ of the driving part and the non-contact posture ϕ of the passive part, currently obtained, are registered so that the posture ϕ of the driving part is moved by $\Delta\phi$ (step S2204). The arithmetic operation of the non-contact posture ϕ of the passive part with respect to the driving part moved at pitches of $\Delta\phi$ (step S2202), and the process of registering the postures ϕ, ϕ (step S2203), are repeated until the posture ϕ of the driving part assumes the final posture.

[0280]

FIG. 53 is a flow diagram illustrating a non-contact

posture retrieving routine, which is to be executed by step S2202 of the cam-relation retrieving routine of FIG. 52, for retrieving the non-contact posture of the passive part.

[0281]

In step S2301, the current posture ϕ of the passive part is set to ϕ_0 so that the initial value 0 is set for the count value n of a counter.

[0282]

In step S2302, it is judged whether or not the driving part and the passive part interfere with each other; if there is no interference, the current posture ϕ is regarded as a non-contact posture (step S2303).

[0283]

If it is judged in step S2302 that interference has occurred, the procedure advances to step S2303 where the count value n is incremented by 1, and then advances to step S2304 where $\phi_0+n\Delta\phi$ is set as the posture ϕ of the passive part. It is judged again whether or not there is present interference (step S2305). If interference is present, the procedure advances to step S2306 where $\phi_0-n\Delta\phi$ is set as the posture ϕ of the passive part. It is judged again whether or not there is present interference (step S2307). If interference is present, the procedure advances to step S2308 where it is judged whether or not the count value n is the upper limit of n. If it is the upper limit of n, it is judged that the retrieval is unsuccessful (step S2309). If the count value n has not yet reached the upper limit, the procedure returns to step S2303 where the count value

n is incremented by 1. In the same manner, an interference check is performed.

[0284]

If it is judged in step S2305 that there is no interference, the procedure advances to step S2310 where ϕ is decremented by $\Delta\phi$, and $\Delta\phi$ is divided by a suitable integer N and is set to a smaller value. Then, while ϕ is being incremented by $\Delta\phi$, which has been set to the smaller value as mentioned above (step S2313), it is judged whether or not there is interference (step S2312). If the interfered state has been changed to the no-interfered state, the procedure advances to step S2314 where it is judged whether or not retrieval has been performed with a predetermined degree of precision. If the precision has not reached the predetermined precision, the procedure returns to step S2310 where $\Delta\phi$ is subtracted from ϕ, and then advances to step S2311 where an even smaller value is set as $\Delta\phi$, whereupon an interference check is performed while ϕ is being incremented by the even smaller value $\Delta\phi$ (steps S2312 and S2313).

[0285]

Thus, if retrieval is performed with a predetermined degree of precision (step S2314), a non-contact posture is thereby determined (step S2303).

[0286]

When it is judged in step S2307 that there is no interference, the same steps as steps S2310 through S2314 are executed so that a non-contact posture of the passive part is obtained. When it is judged in step S2307 that no interference

is present, ϕ is incremented by $\Delta\phi$ in a step corresponding to step S2310, and ϕ is decremented by $\Delta\phi$ in a step corresponding to step S2313.

[0287]

FIG. 54 is a flow diagram illustrating a non-contact posture retrieving routine, which is to be executed by step S2202 of the cam-relation retrieving routine of FIG. 52, for retrieving a non-contact posture of the passive part. The routine of FIG. 54 is a routine to be executed when gravity or spring force acts on the passive part. The direction in which gravity or spring force acts on the passive part is previously set by the user. FIG. 53 shows the case where force is exerted in a negative direction.

[0288]

In the routine of FIG. 54, in step S2401 the current posture ϕ of the passive part is set to ϕ_0, and the count value n of a counter is set to an initial value 0.

[0289]

In step S2402, it is judged whether or not the driving part and the passive part currently interfere with each other. When there is interference, the current posture change in the driving part interferes with the passive part. Therefore, the posture of the passive part is changed against gravity or spring force (steps S2404 through s2407). After a posture with no interference has been found, $\Delta\phi$ is subtracted from ϕ and the passive part is temporarily returned to the interfering posture (step S2408).

[0290]

On the other hand, if it is judged in step S2402 that there is no interference, the current posture change in the driving part does not interfere with the passive part. Therefore, the effect of gravity or spring force is simulated to find a state in which the passive part interfere with the positive part (steps S2409 through S2413).

[0291]

When the procedure advances to step S2414, interference has occurred with a width of $\Delta\phi$. Therefore, $\Delta\phi$ is divided by N and a smaller value is set as $\Delta\phi$. After that, a non-interference posture is retrieved while ϕ is being incremented by $\Delta\phi$ (steps S2415 and S2416). In step S2417, it is judged whether retrieval has been performed up to a predetermined precision. If the precision has not reached the predetermined precision, the procedure advances to step S2408 where $\Delta\phi$ is subtracted from ϕ to restore the interfering posture, and then advances to step S2414 where $\Delta\phi$ is divided by an integer N. An even smaller value is set as $\Delta\phi$, and retrieval is repeated.

[0292]

Thus, if retrieval has been performed up to a predetermined degree of precision (step S2414), a non-contact position is determined (step S2418).

[0293]

In this manner, the cam relationship can be determined even when gravity or spring force is exerted or not exerted.

[0294]

FIG. 55 is an explanatory diagram illustrating how groove relationship with a pin inserted into a groove is set.

[0295]

When the groove relationship of FIG. 55 is set, the pin is normally in contact with the groove. Therefore, a non-interference position cannot be obtained, and such groove relationship is difficult to obtain by the aforementioned cam-relation retrieving method.

[0296]

Hence, as shown in FIG. 55C, a very small cylindrical tube having a radius r is temporarily attached to the center of a pin, and a distance d between the cylindrical tube and the groove is measured. With the measurement, groove relationship is obtained. That is, if judgement of "d<R-r?" is made instead of the judgement of interference in FIG. 53, the groove relationship can be retrieved.

[0297]

[Advantages of the Invention]

According to the present invention, as described above, a program for controlling a mechanism can be debugged without physically generating the mechanism. This will make a large contribution to a reduction in time and cost of the development of the mechanism control program.

[Brief Description of the Drawings]

[FIG. 1]

FIG. 1 is a diagram illustrating the exterior of a computer system in which a mechanism-control-program development support system is constructed.

[FIG. 2]

FIG. 2 is a block diagram illustrating the hardware of the computer system shown in FIG 1.

[FIG. 3]

FIG. 3 is a diagram illustrating the construction of programs stored on a program storage medium.

[FIG. 4]

FIG. 4 is a diagram illustrating an example of a computer network where an example of the mechanism-control-program development support system of the present invention is constructed.

[FIG. 5]

FIG. 5 is a block diagram illustrating the hardware of the computer system constituting the computer network shown in FIG. 4.

[FIG. 6]

FIG. 6 is a conceptual diagram of the mechanism-control-program development support system.

[FIG. 7]

FIG. 7 is a schematic diagram illustrating how a 3D model simulator processes a sensor and a motor when defining them.

[FIG. 8]

FIG 8 is a schematic diagram illustrating a simulation process that is performed by the 3D model simulator.

[FIG. 9]

FIG. 9 is a timing diagram illustrating the operating timing, during simulation, between the 3D model simulator and the control program.

[FIG. 10]

FIG. 10 is a flow diagram illustrating the operation of the simulator at the time of simulation.

[FIG. 11]

FIG. 11 is a flow diagram illustrating a partial flow (a portion enclosed by a dot-and-dash line in FIG. 10) that can be substituted for part of the process flow shown in FIG. 10.

[FIG. 12]

FIG. 12 is a schematic diagram illustrating how to select a link that is defined as a motor.

[FIG. 13]

FIG. 13 is a schematic diagram illustrating how to select a link that is defined as a sensor.

[FIG. 14]

FIG. 14 is a schematic diagram illustrating how to select a link that is defined as a sensor.

[FIG. 15]

FIG. 15 is a schematic diagram illustrating motor types.

[FIG. 16]

FIG. 16 is a graph illustrating an initial change in the rotation rate ω of a motor when the motor is regarded as a primary delay system.

[FIG. 17]

FIG. 17 is a flow diagram illustrating how a motor driving process in the simulator is performed with a primary delay considered.

[FIG. 18]

FIG. 18 is an explanatory diagram illustrating how to define a sensor using an interference check.

[FIG. 19]

FIG. 19 is an explanatory diagram illustrating how to define a sensor using an interference check.

[FIG. 20]

FIG. 20 is an explanatory diagram illustrating how to define a sensor using an interference check.

[FIG. 21]

FIG. 21 is a timing diagram illustrating signals that are transmitted and received for obtaining synchronization between the simulator and the control program.

[FIG. 22]

FIG. 22 is a flow diagram illustrating a synchronizing program for obtaining synchronization with the movement of the 3D model.

[FIG. 23]

FIG. 23 is a flow diagram illustrating the process that is performed by the simulator to realize synchronization.

[FIG. 24]

FIG. 24 is an explanatory diagram illustrating how to evaluate the possibility of interference.

[FIG. 25]

FIG. 25 is a graph illustrating various kinds of relationships between the inter-part distance d and the simulation interval Δt.

[FIG. 26]

FIG. 26 is an explanatory diagram of grouping of parts.

[FIG. 27]

FIG. 27 is an explanatory diagram illustrating grouping in the case where a plurality of motor links are defined in a single three-dimensional model.

[FIG. 28]

FIG. 28 is an explanatory diagram illustrating regrouping of the grouped parts.

[FIG. 29]

FIG. 29 is a flow diagram illustrating the routine of dividing a plurality of links into a group of links which are affected in posture by movement of the motor link and a group of links which are not affected in posture by the movement of the motor link.

[FIG. 30]

FIG. 30 is a flow diagram illustrating a retrieval routine.

[FIG. 31]

FIG. 31 is a diagram illustrating a link mechanism model for explaining operation of the routines of FIGS. 29 and 30.

[FIG. 32]

FIG. 32 is a diagram illustrating a data structure for the link mechanism model shown in FIG. 31.

[FIG. 33]

FIG. 33 is an explanatory diagram illustrating a method of evaluating the possibility of interference in the group of links whose postures are changed by movement of a single motor link.

[FIG. 34]

FIG. 34 is an explanatory diagram illustrating a method of extracting, from the links affected in posture by movement of the motor link, a set of links between which distance is calculated during simulation.

[FIG. 35]

FIG. 35 is a flow diagram illustrating the entire simulation process of this embodiment.

[FIG. 36]

FIG. 36 is a flow diagram illustrating a motor processing routine.

[FIG. 37]

FIG. 37 is a flow diagram illustrating a link posture movement subroutine.

[FIG. 38]

FIG. 38 is a flow diagram illustrating a joint malfunction processing routine.

[FIG. 39]

FIG. 39 is a flow diagram illustrating a sensor processing routine.

[FIG. 40]

FIG. 40 is a flow diagram illustrating an ON/OFF sensor

processing routine.

[FIG. 41]

FIG. 41 is a flow diagram illustrating a sensor performance degradation processing routine.

[FIG. 42]

FIG. 42 is a flow diagram illustrating an estimated output value change processing routine for chattering.

[FIG. 43]

FIG. 43 is a graph showing one example of an output waveform for movement (e.g., angle) of an encoder.

[FIG. 44]

FIG. 44 is a diagram showing various output waveforms of the encoder.

[FIG. 45]

FIG. 45 is a diagram illustrating a program of an output processing routine for the encoder.

[FIG. 46]

FIG. 46 is a flow diagram illustrating an encoder processing routine.

[FIG. 47]

FIG. 47 is a diagram illustrating function values for a potentiometer.

[FIG. 48]

FIG. 48 is a flow diagram illustrating a potentiometer preprocessing routine.

[FIG. 49]

FIG. 49 is a flow diagram illustrating a ΔT determination

processing routine.

[FIG. 50]

FIG. 50 is an explanatory diagram illustrating a method, based on an interference check, of retrieving a joint movable range.

[FIG. 51]

FIG. 51 is an explanatory diagram illustrating how a cam relation is retrieved by an interference check.

[FIG. 52]

FIG. 52 is a flow diagram illustrating a cam relation retrieving routine based on an interference check.

[FIG. 53]

FIG. 53 is a flow diagram illustrating a non-contact posture retrieving routine for retrieving the non-contact posture of a passive part.

[FIG. 54]

FIG. 54 is a flow diagram illustrating a non-contact posture retrieving routine for retrieving the non-contact posture of a passive part.

[FIG. 55]

FIG. 55 is an explanatory diagram illustrating how groove relationship with a pin inserted into a groove is set.

[Description of Reference Numerals]

100, 400, 500 ... Computer system

101, 401, 501 ... Main body portion

101a, 401a, 501a ... Floppy disk loading slot

101b, 401b, 501b ... CDROM loading slot

102, 402, 502 ... CRT display

102a, 402a, 502a ... Display screen

103, 403, 503 ... Keyboard

104, 404, 504 ... Mouse

210, 410 ... CDROM

211, 411 ... Hard disk

212, 412 ... Floppy disk

220, 420 ... Bus

221, 421 ... Central processing unit (CPU)

222, 422 ... RAM

223, 423 ... Hard disk control

224, 424 ... Floppy disk driver

225, 425 ... CDROM driver

226, 426 ... Mouse controller

227, 427 ... Keyboard controller

228, 428 ... Display controller

300 ... Program storage medium

310 ... Mechanism-control-program development support program

311 ... Three-dimensional mechanism model simulation program

312 ... Synchronizing program

429 ... Modem

[Tittle of Document] Abstract

[Abstract]

[Problem]

To support the development of a control program for controlling a mechanism.

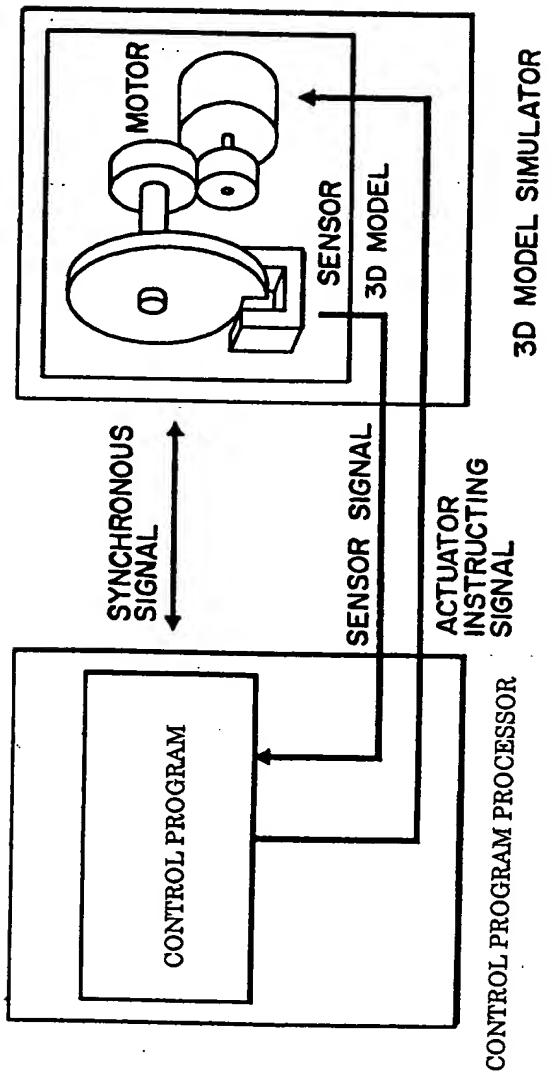
[Means for Solution]

A mechanism-control-program development support system comprising: a three-dimensional mechanism model simulator for operating a three-dimensional mechanism model, constructed within the simulator, which is composed of a plurality of parts including actuators and sensors; and a control program processor for executing a control program for controlling operation of the three-dimensional mechanism model constructed within the three-dimensional mechanism model simulator, while synchronizing the control program with the three-dimensional mechanism model which operates within the three-dimensional mechanism model simulator.

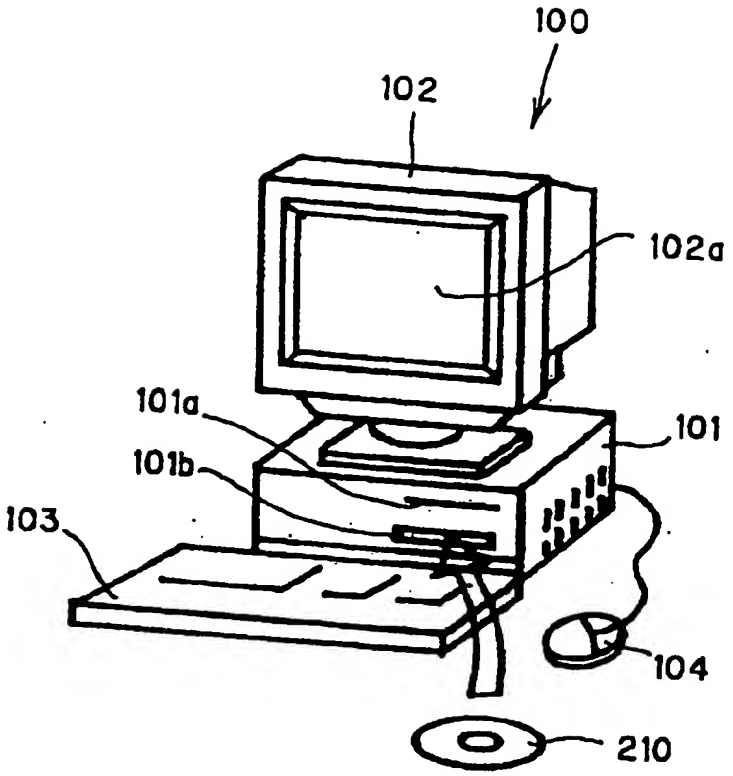
[Selected Drawing] Figure 6

[Name of Document] Drawings

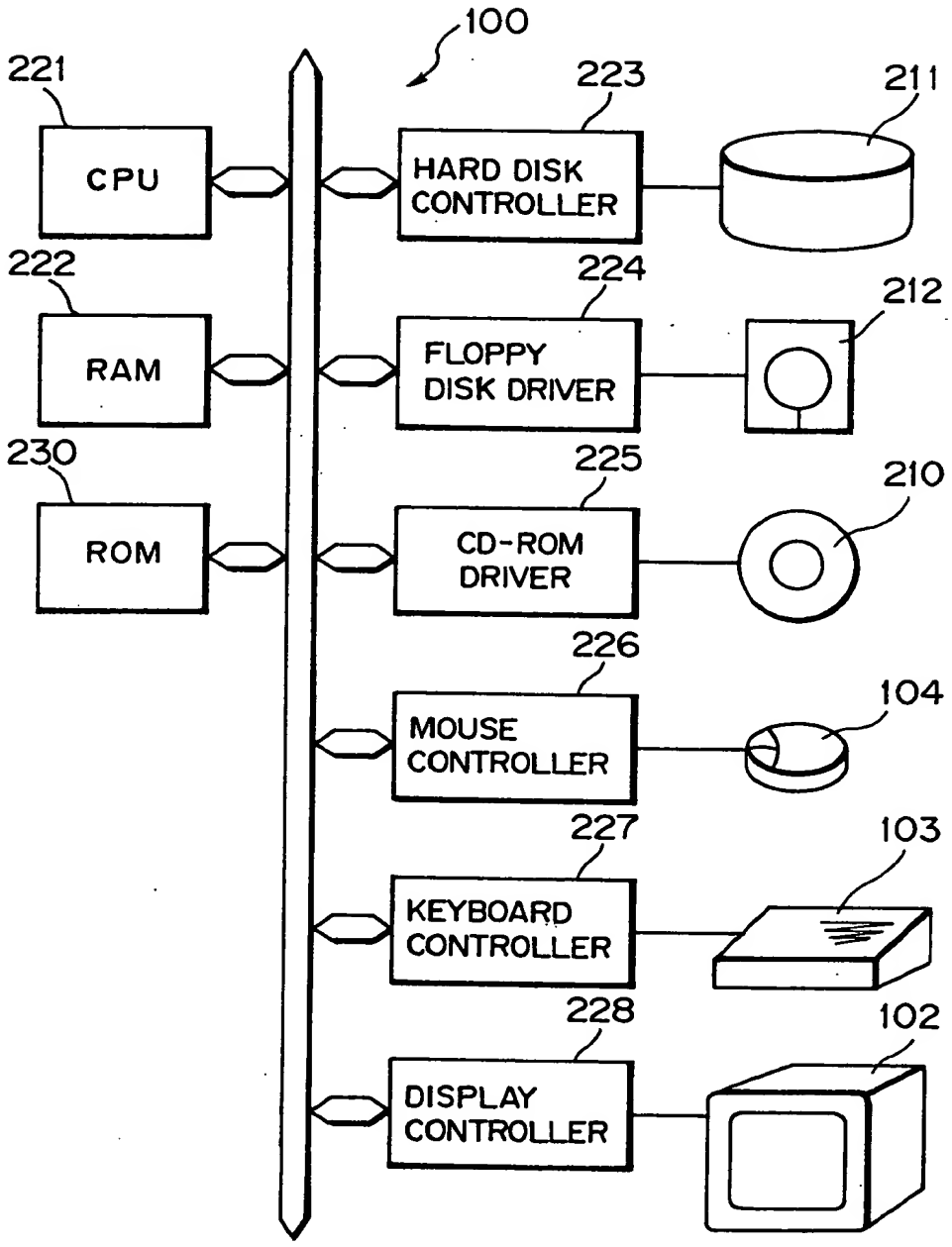
[FIG. 6]



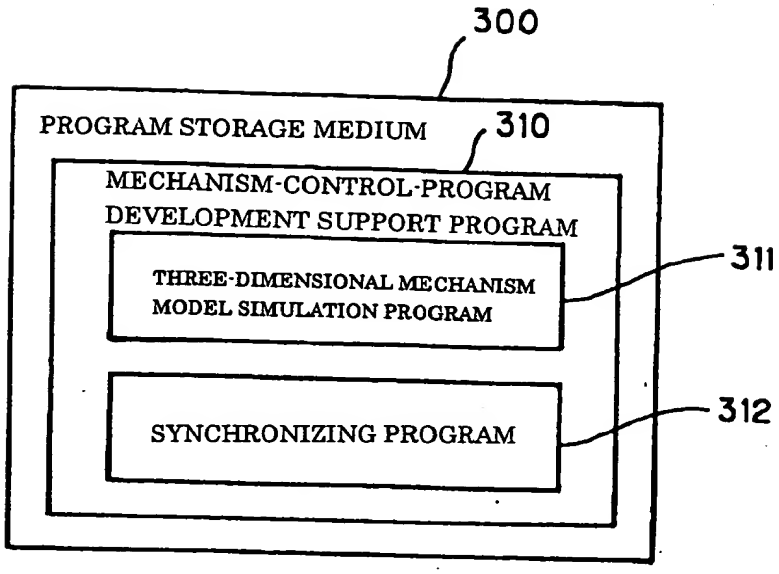
[FIG. 1]



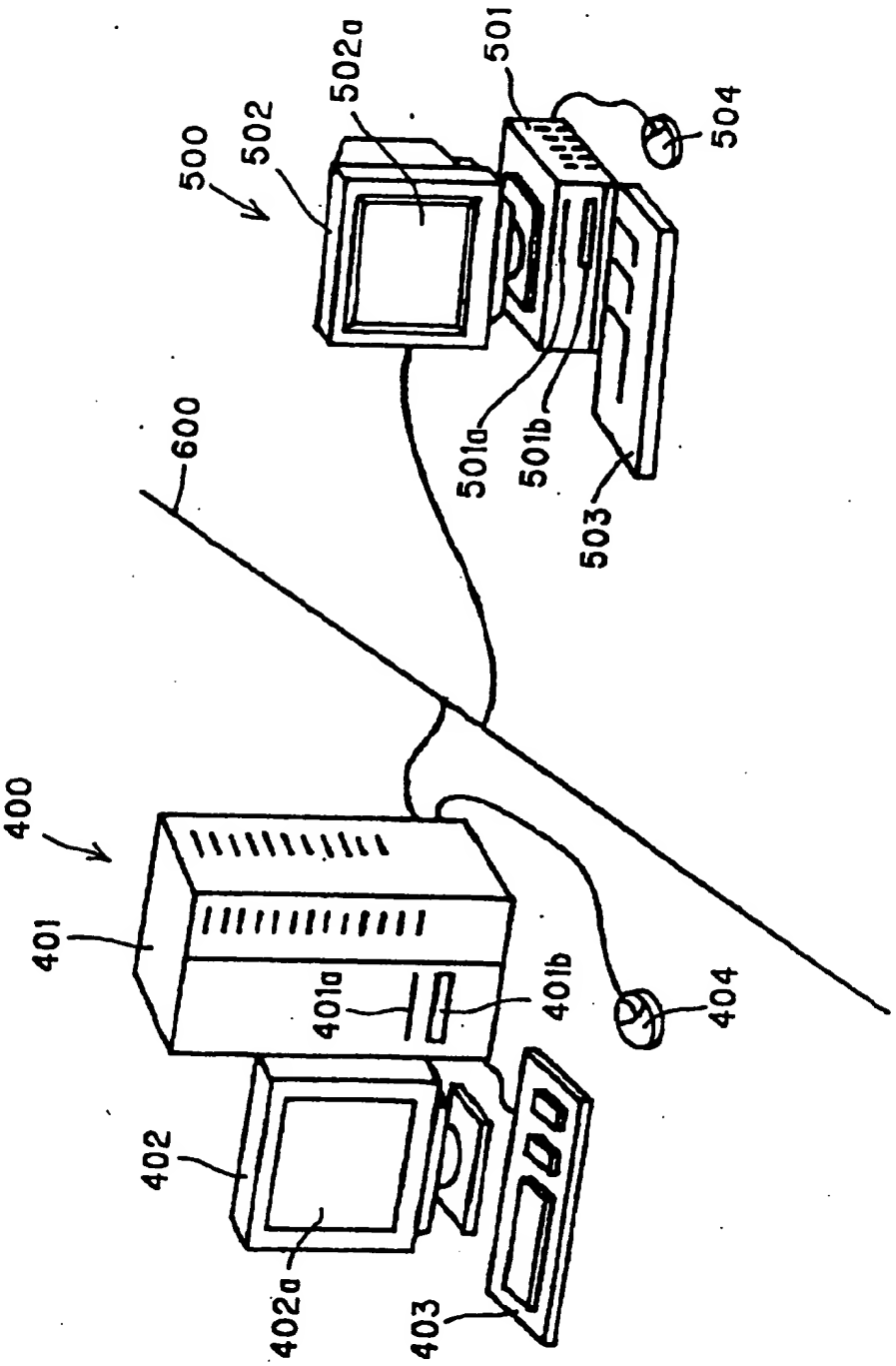
[FIG. 2]


100
221
CPU
222
RAM
230
ROM
223
HARD DISK CONTROLLER
224
FLOPPY DISK DRIVER
225
CD-ROM DRIVER
226
MOUSE CONTROLLER
227
KEYBOARD CONTROLLER
228
DISPLAY CONTROLLER
211
212
210
104
103
102

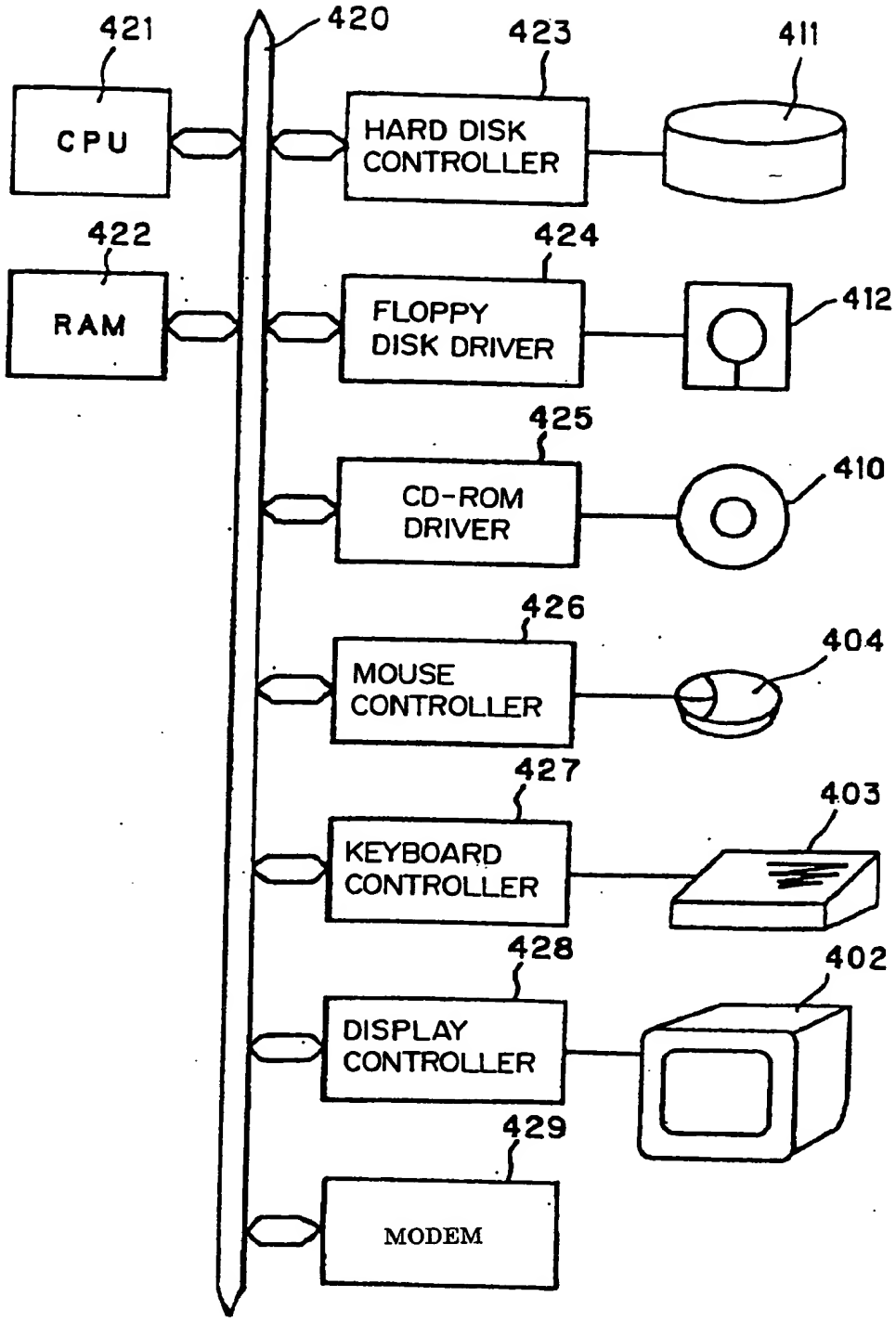
[FIG. 3]


300
PROGRAM STORAGE MEDIUM
310
MECHANISM-CONTROL-PROGRAM
DEVELOPMENT SUPPORT PROGRAM
THREE-DIMENSIONAL MECHANISM
MODEL SIMULATION PROGRAM
311
SYNCHRONIZING PROGRAM
312

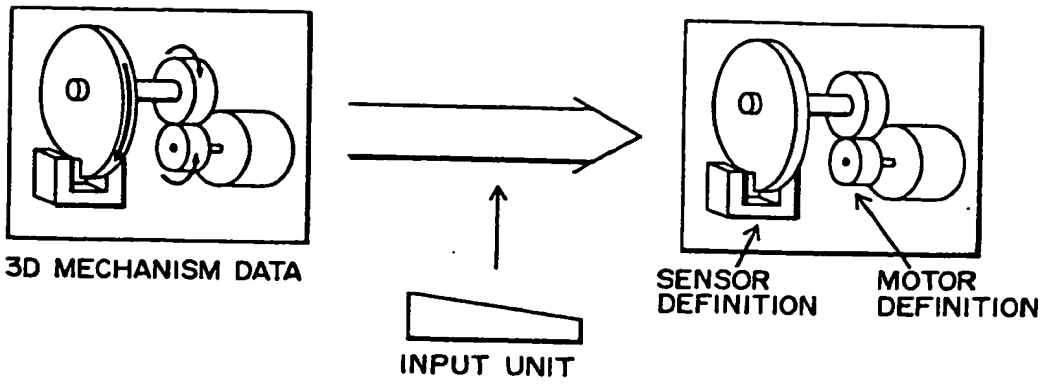
[FIG. 4]


500
502
502a
501
504
600
501a
501b
503
400
401
401a
401b
404
402
402a
403

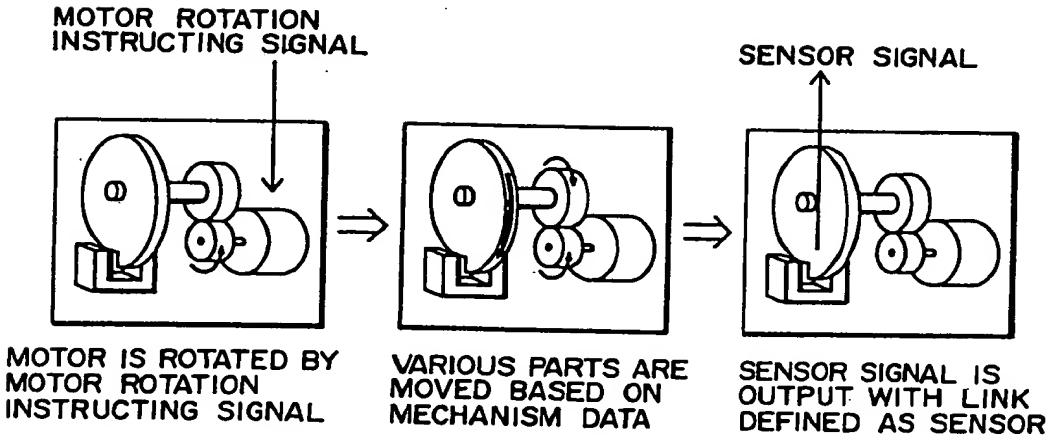
[FIG. 5]


421
420
423
411
CPU
HARD DISK CONTROLLER
422
424
RAM
FLOPPY DISK DRIVER
412
425
CD-ROM DRIVER
410
426
MOUSE CONTROLLER
404
427
403
KEYBOARD CONTROLLER
428
402
DISPLAY CONTROLLER
429
MODEM

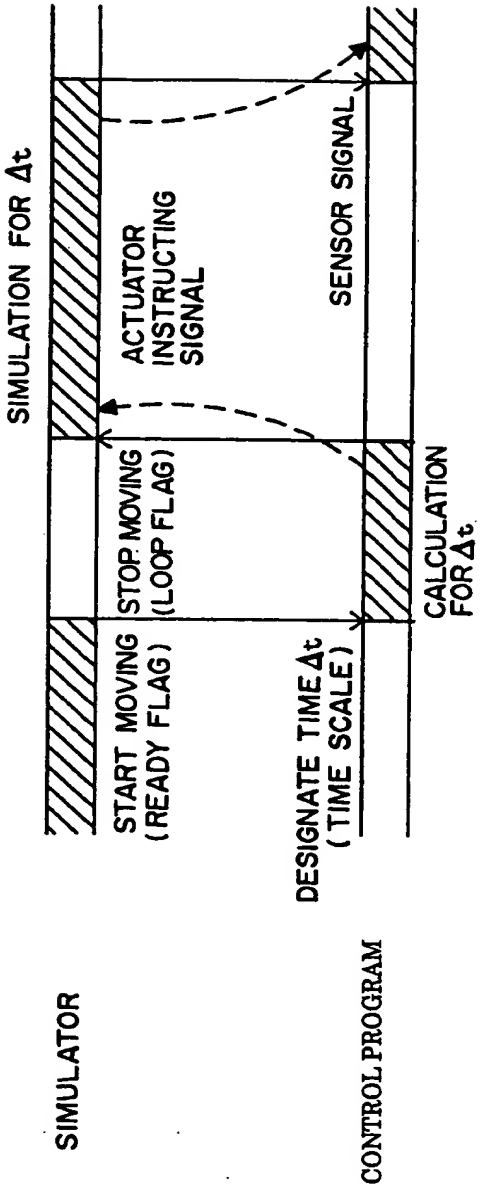
[FIG. 7]


3D MECHANISM DATA
INPUT UNIT
SENSOR DEFINITION
MOTOR DEFINITION

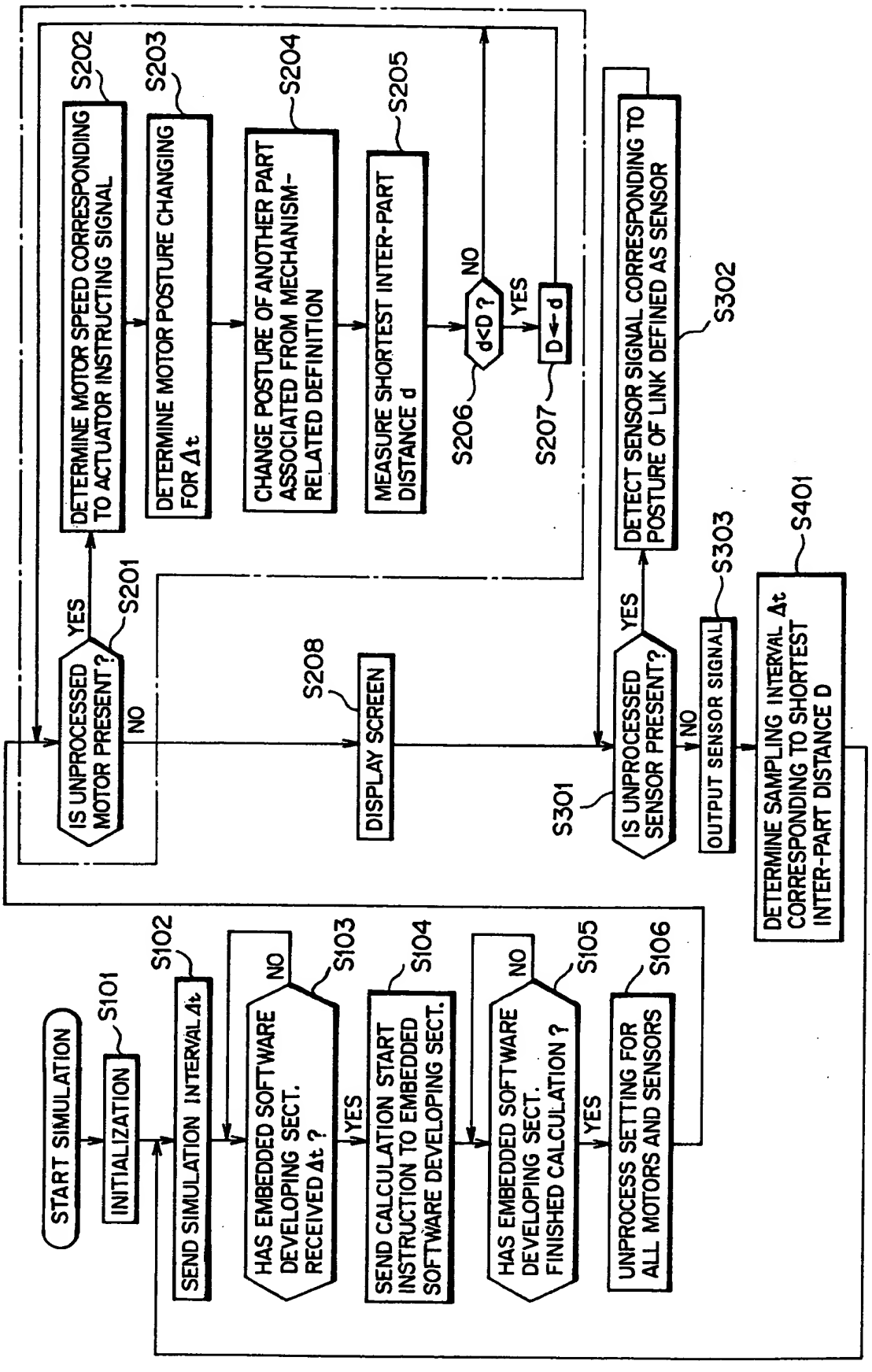
[FIG. 8]


MOTOR ROTATION INSTRUCTING SIGNAL
SENSOR SIGNAL
MOTOR IS ROTATED BY MOTOR ROTATION INSTRUCTING SIGNAL
VARIOUS PARTS ARE MOVED BASED ON MECHANISM DATA
SENSOR SIGNAL IS OUTPUT WITH LINK DEFINED AS SENSOR

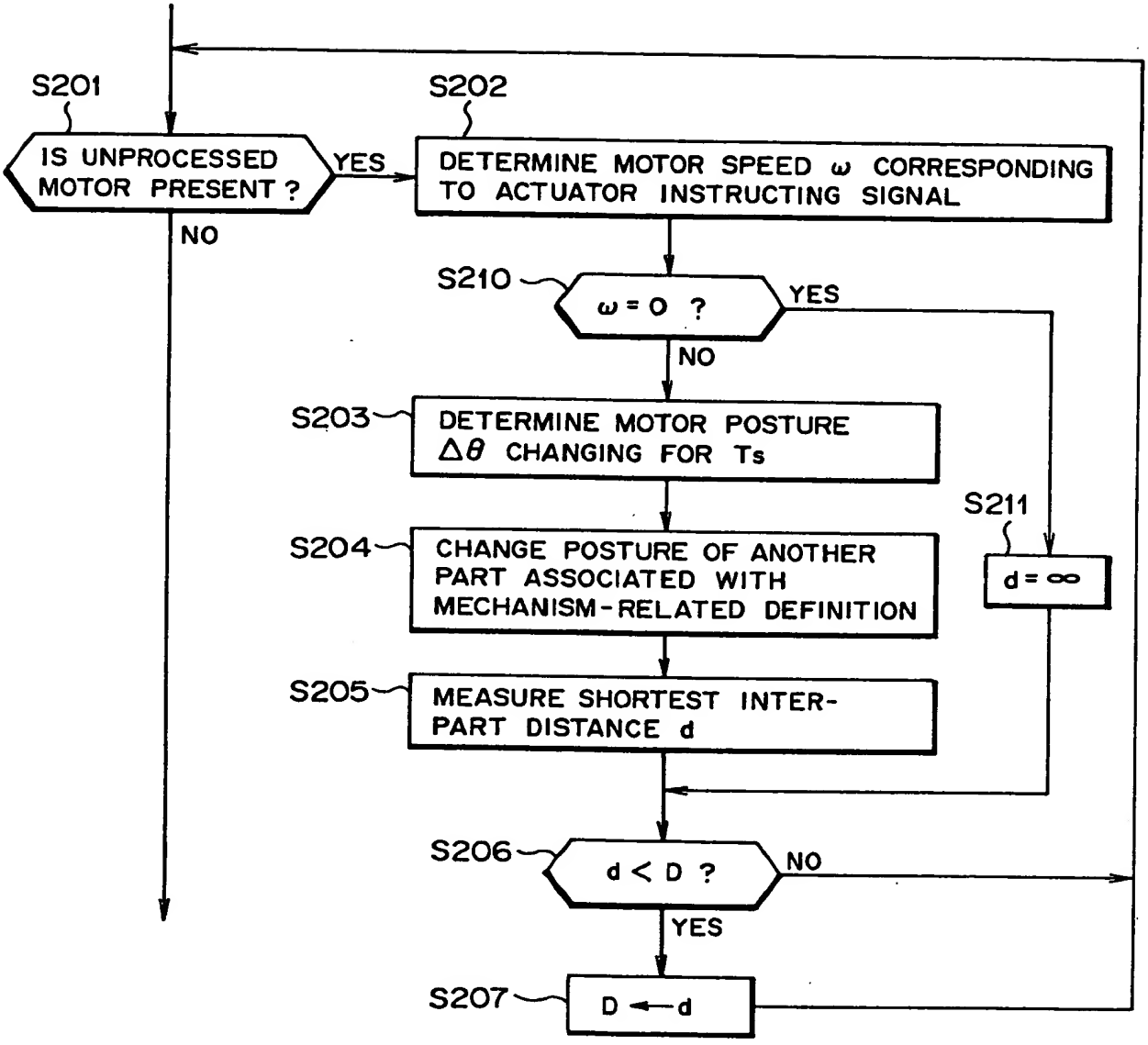
[FIG. 9]


SIMULATION FOR Δt
ACTUATOR INSTRUCTING SIGNAL
SENSOR SIGNAL
STOP MOVING (LOOP FLAG)
START MOVING (READY FLAG)
DESIGNATE TIME Δt (TIME SCALE)
CALCULATION FOR Δt
SIMULATOR
CONTROL PROGRAM

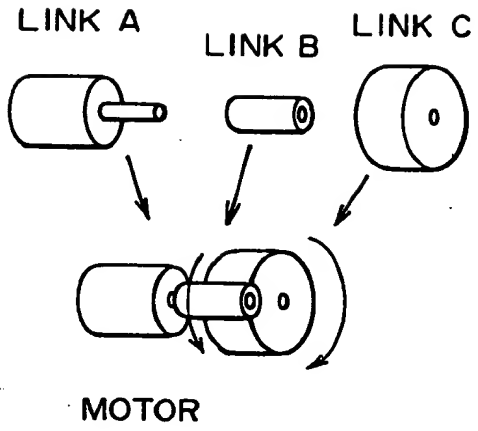
[FIG. 10]


START SIMULATION
INITIALIZATION
S101
SEND SIMULATION INTERVAL Δt
S102
HAS EMBEDDED SOFTWARE DEVELOPING SECT. RECEIVED Δt ?
S103
NO
YES
SEND CALCULATION START INSTRUCTION TO EMBEDDED SOFTWARE DEVELOPING SECT.
S104
HAS EMBEDDED SOFTWARE DEVELOPING SECT. FINISHED CALCULATION ?
S105
NO
YES
UNPROCESS SETTING FOR ALL MOTORS AND SENSORS
S106
IS UNPROCESSED MOTOR PRESENT?
S201
YES
NO
DETERMINE MOTOR SPEED CORRESPONDING TO ACTUATOR INSTRUCTING SIGNAL
S202
DETERMINE MOTOR POSTURE CHANGING FOR Δt
S203
CHANGE POSTURE OF ANOTHER PART ASSOCIATED FROM MECHANISM-RELATED DEFINITION
S204
MEASURE SHORTEST INTER-PART DISTANCE d
S205
d<D ?
S206
NO
YES
D←d
S207
DISPLAY SCREEN
S208
IS UNPROCESSED SENSOR PRESENT?
S301
YES
NO
DETECT SENSOR SIGNAL CORRESPONDING TO POSTURE OF LINK DEFINED AS SENSOR
S302
OUTPUT SENSOR SIGNAL
S303
DETERMINE SAMPLING INTERVAL Δt CORRESPONDING TO SHORTEST INTER-PART DISTANCE D
S401

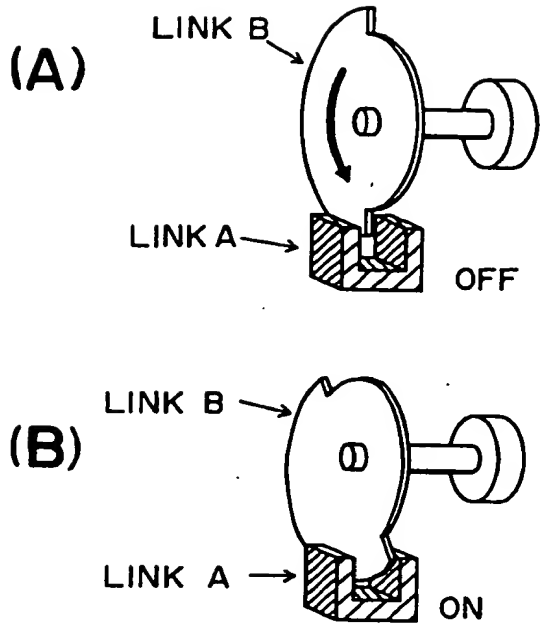
[FIG. 11]


S201
IS UNPROCESSED MOTOR PRESENT ?
YES
NO
S202
DETERMINE MOTOR SPEED ω CORRESPONDING TO ACTUATOR INSTRUCTING SIGNAL
S210
ω = 0 ?
YES
NO
S203
DETERMINE MOTOR POSTURE Δθ CHANGING FOR Ts
S204
CHANGE POSTURE OF ANOTHER PART ASSOCIATED WITH MECHANISM-RELATED DEFINITION
S205
MEASURE SHORTEST INTER-PART DISTANCE d
S211
d = ∞
S206
d < D ?
NO
YES
S207
D ← d

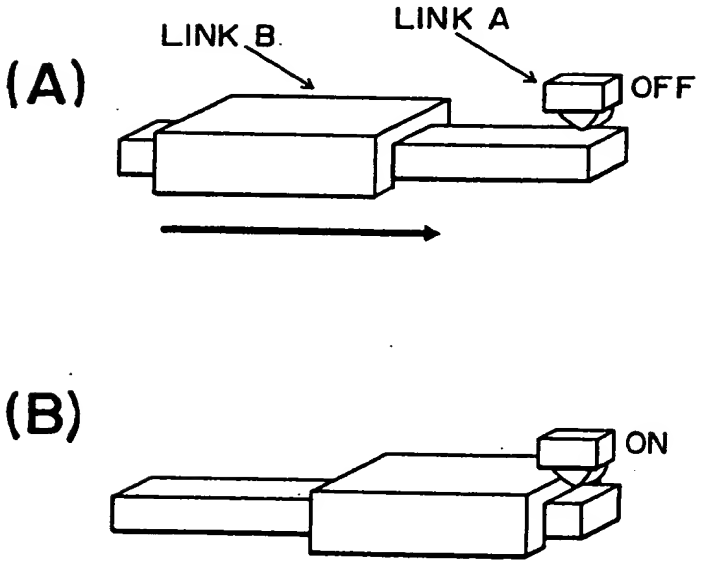
[FIG. 12]



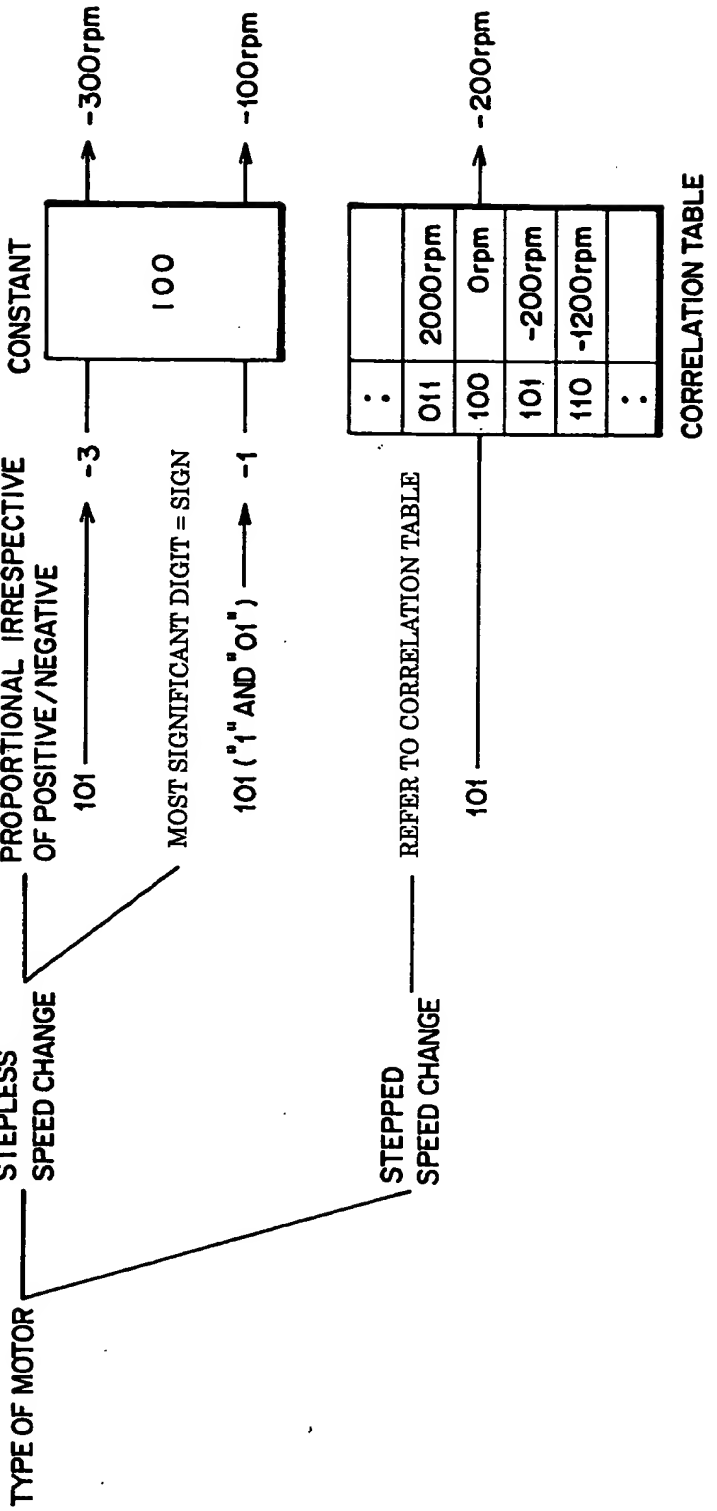
[FIG. 13]



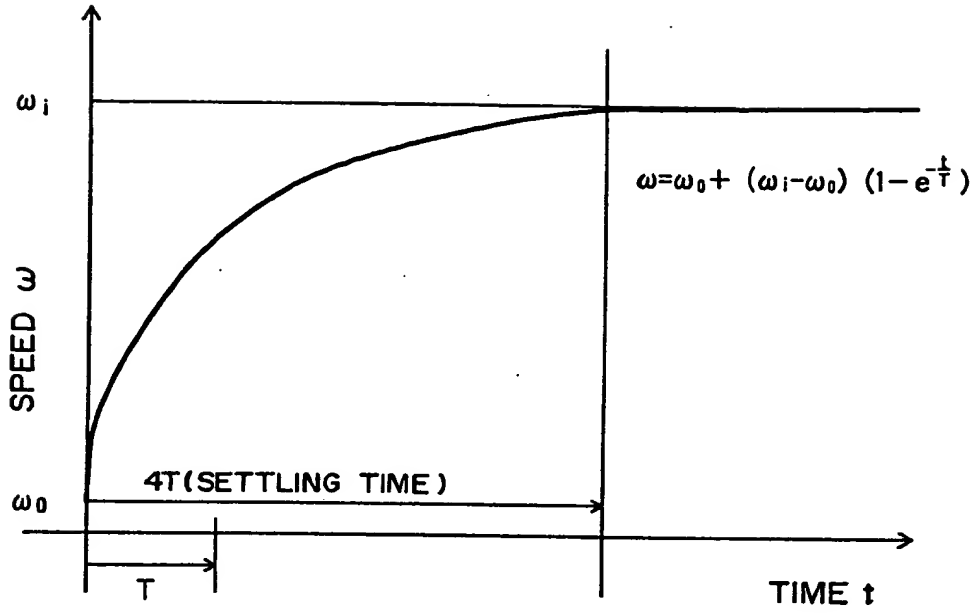
[FIG. 14]



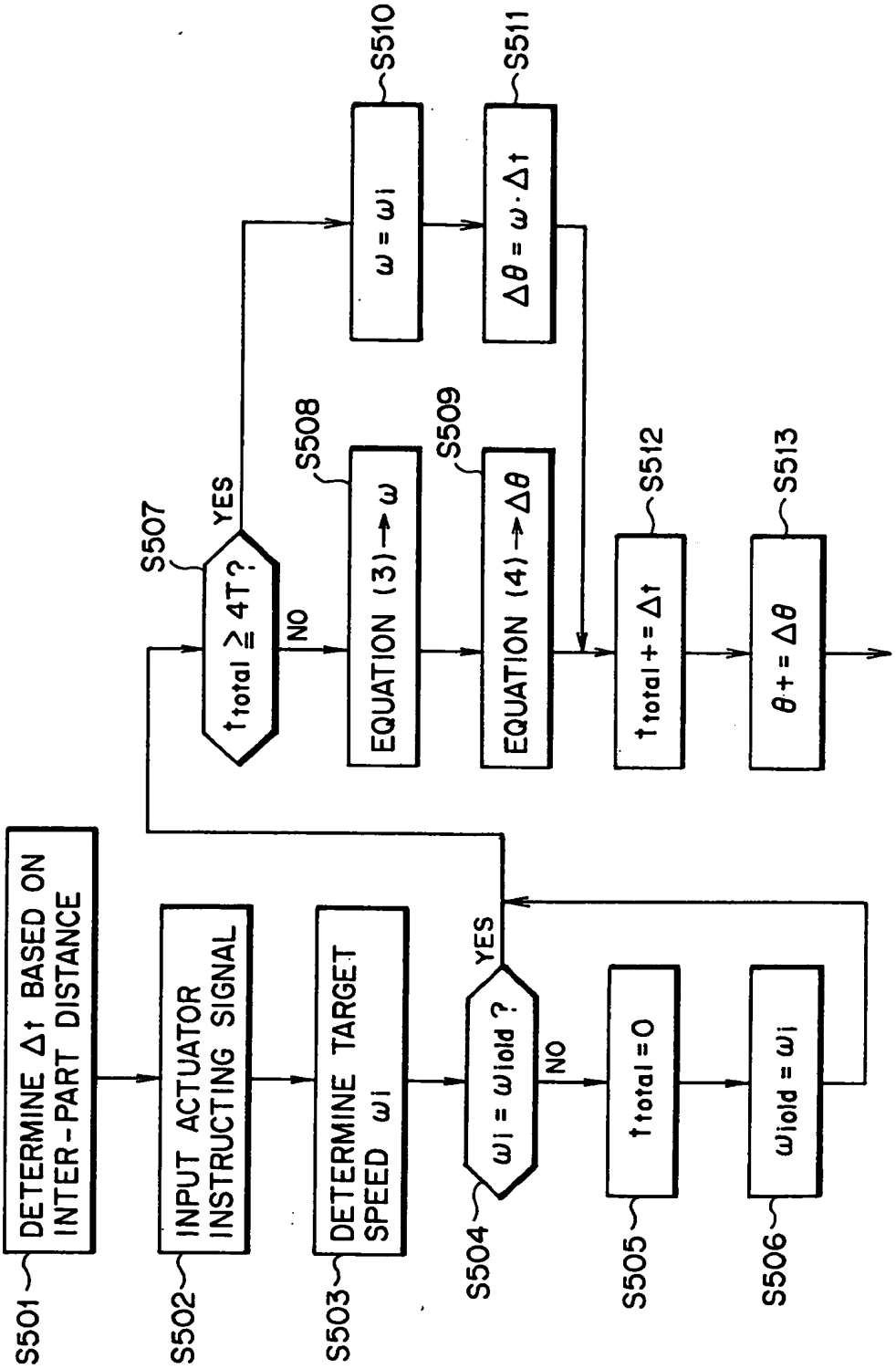
[FIG. 15]


TYPE OF MOTOR
STEPLESS SPEED CHANGE
PROPORTIONAL IRRESPECTIVE OF POSITIVE/NEGATIVE
MOST SIGNIFICANT DIGIT = SIGN
101
101 ("1" AND "01")
-3
-1
PROPORTIONAL CONSTANT
100
-300rpm
-100rpm
STEPPED SPEED CHANGE
REFER TO CORRELATION TABLE
101
-200rpm
CORRELATION TABLE
011 2000rpm
100 0rpm
101 -200rpm
110 -1200rpm

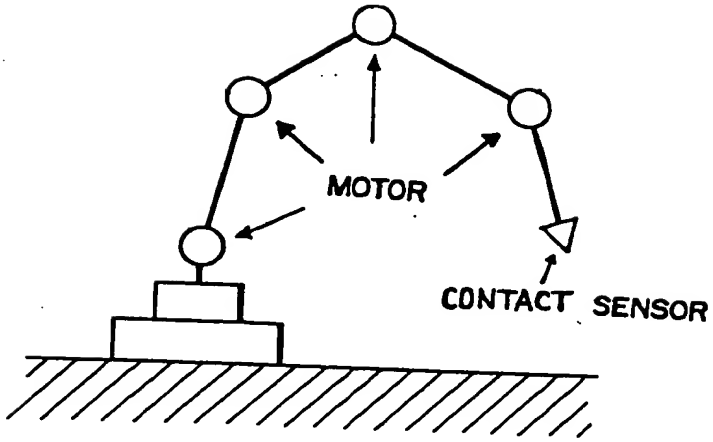
[FIG. 16]


ω_i
ω_0
SPEED ω
$\omega=\omega_0+(\omega_i-\omega_0)(1-e^{-\frac{t}{T}})$
4T(SETTLING TIME)
T
TIME t

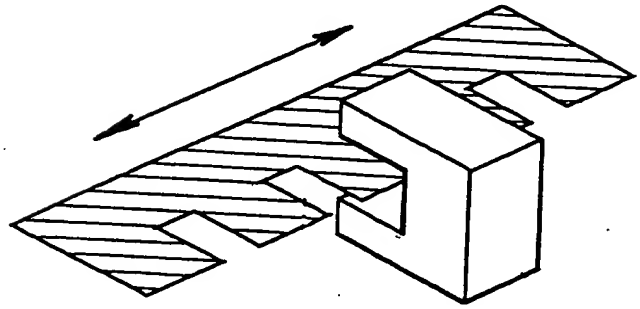
[FIG. 17]


S501
DETERMINE Δt BASED ON INTER-PART DISTANCE
S502
INPUT ACTUATOR INSTRUCTING SIGNAL
S503
DETERMINE TARGET SPEED ωi
S504
ωi = ωiold ?
YES
NO
S505
t total = 0
S506
ωiold = ωi
S507
t total ≧ 4T ?
YES
NO
S508
EQUATION (3) → ω
S509
EQUATION (4) → Δθ
S510
ω = ωi
S511
Δθ = ω · Δt
S512
t total += Δt
S513
θ += Δθ

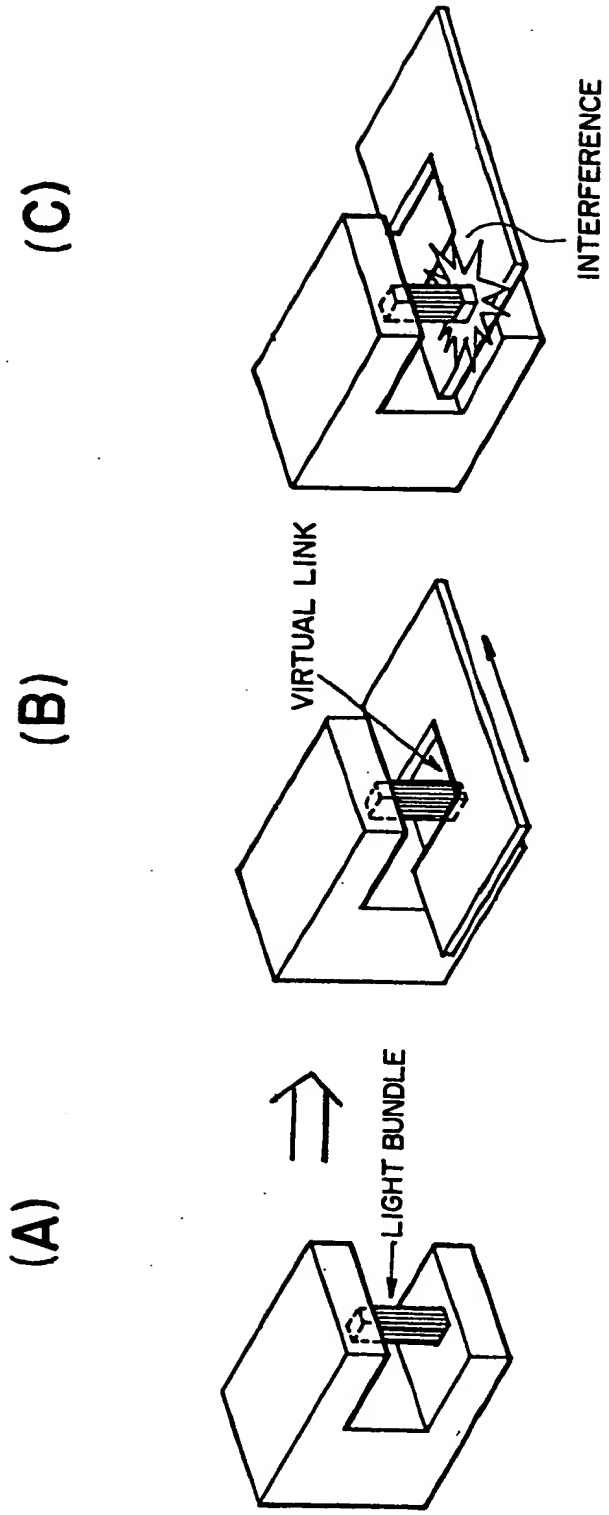
[FIG. 18]



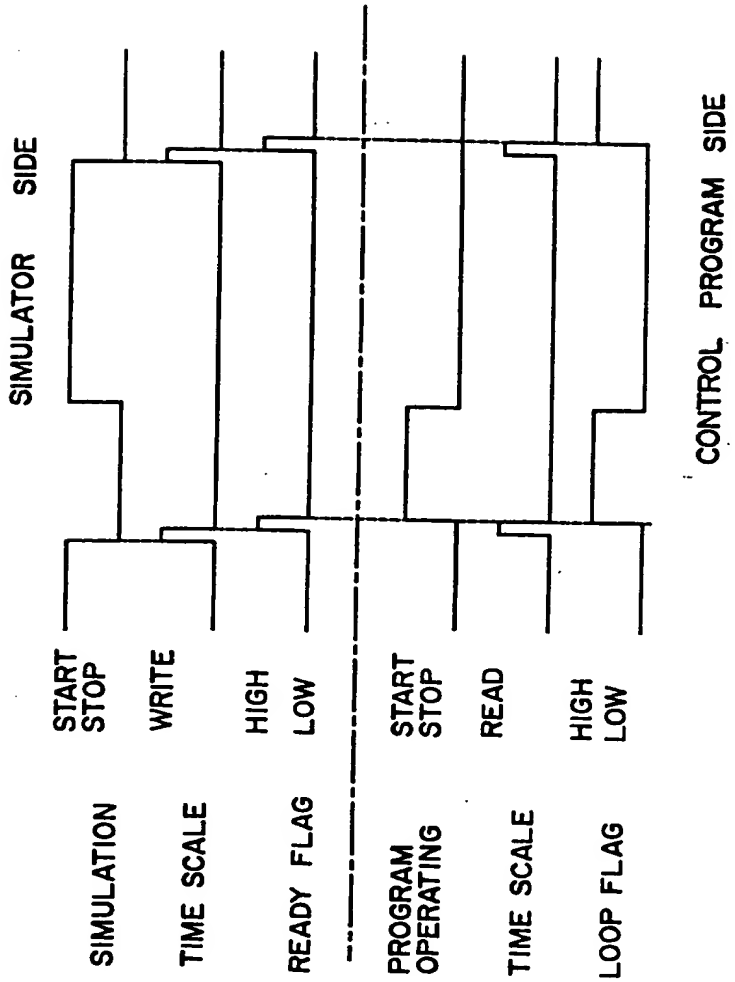
[FIG. 19]



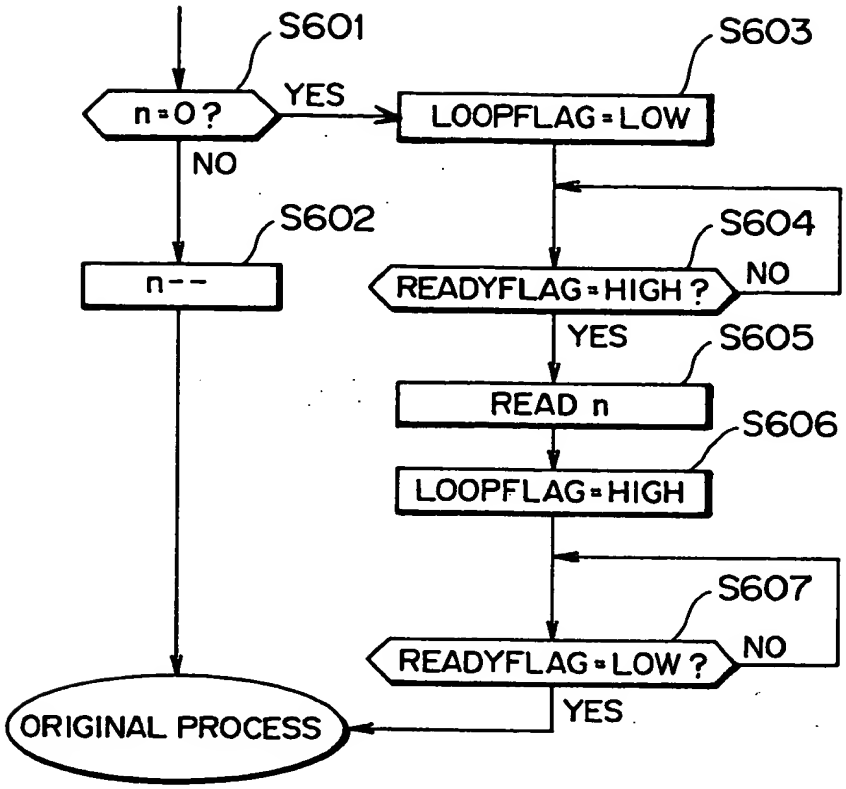
[FIG. 20]



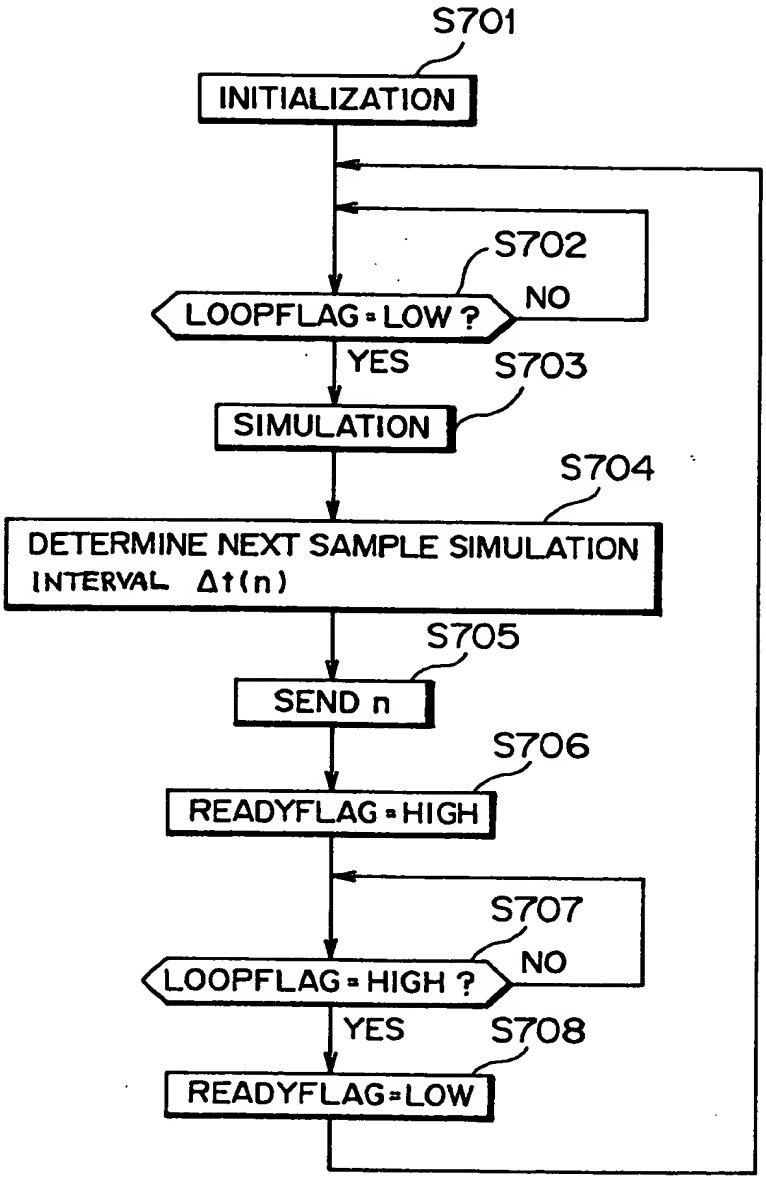
[FIG. 21]


SIMULATOR SIDE
CONTROL PROGRAM SIDE
SIMULATION
START
STOP
TIME SCALE
WRITE
READY FLAG
HIGH
LOW
PROGRAM OPERATING
START
STOP
TIME SCALE
READ
LOOP FLAG
HIGH
LOW

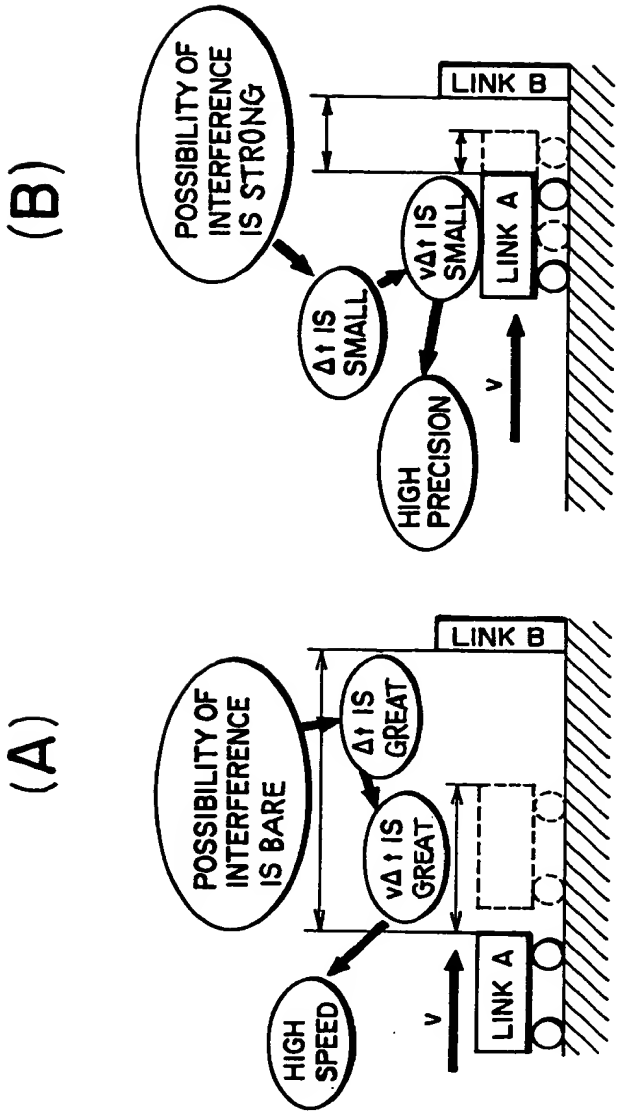
[FIG. 22]


S601
n=0?
YES
NO
S602
n--
S603
LOOPFLAG=LOW
S604
READYFLAG=HIGH?
NO
YES
S605
READ n
S606
LOOPFLAG=HIGH
S607
READYFLAG=LOW?
NO
YES
ORIGINAL PROCESS

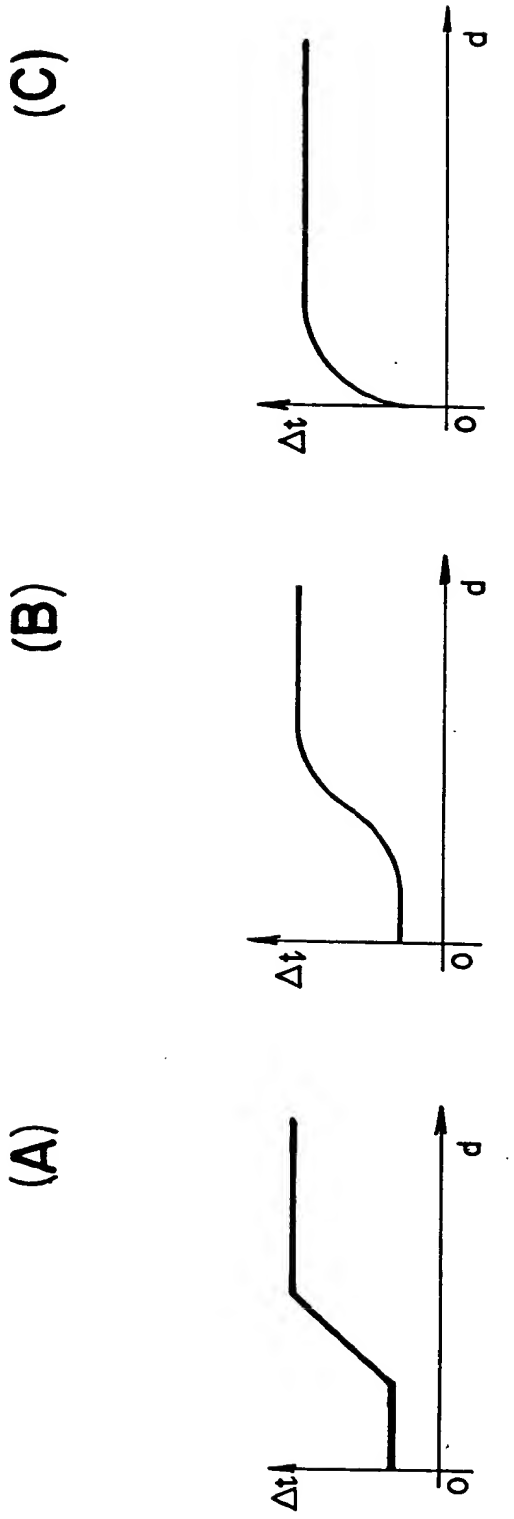
[FIG. 23]


S701
INITIALIZATION
S702
LOOPFLAG = LOW ?
NO
YES
S703
SIMULATION
S704
DETERMINE NEXT SAMPLE SIMULATION INTERVAL Δt(n)
S705
SEND n
S706
READYFLAG = HIGH
S707
LOOPFLAG = HIGH ?
NO
YES
S708
READYFLAG = LOW

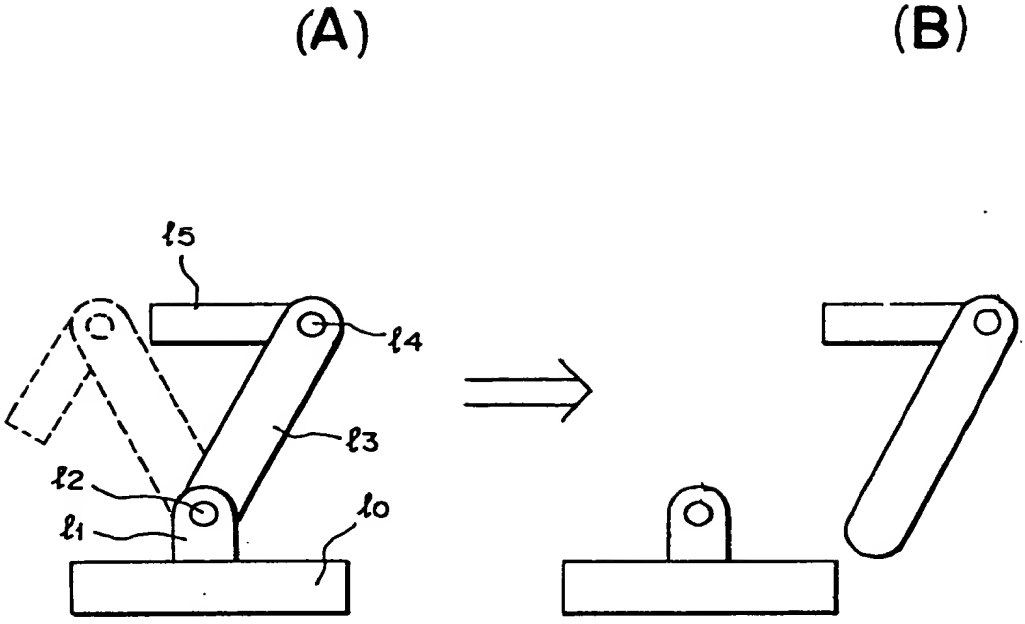
[FIG. 24]


(B)
POSSIBILITY OF INTERFERENCE IS STRONG
Δt IS SMALL
vΔt IS SMALL
LINK B
LINK A
v
HIGH PRECISION
(A)
POSSIBILITY OF INTERFERENCE IS BARE
Δt IS GREAT
vΔt IS GREAT
LINK B
LINK A
v
HIGH SPEED

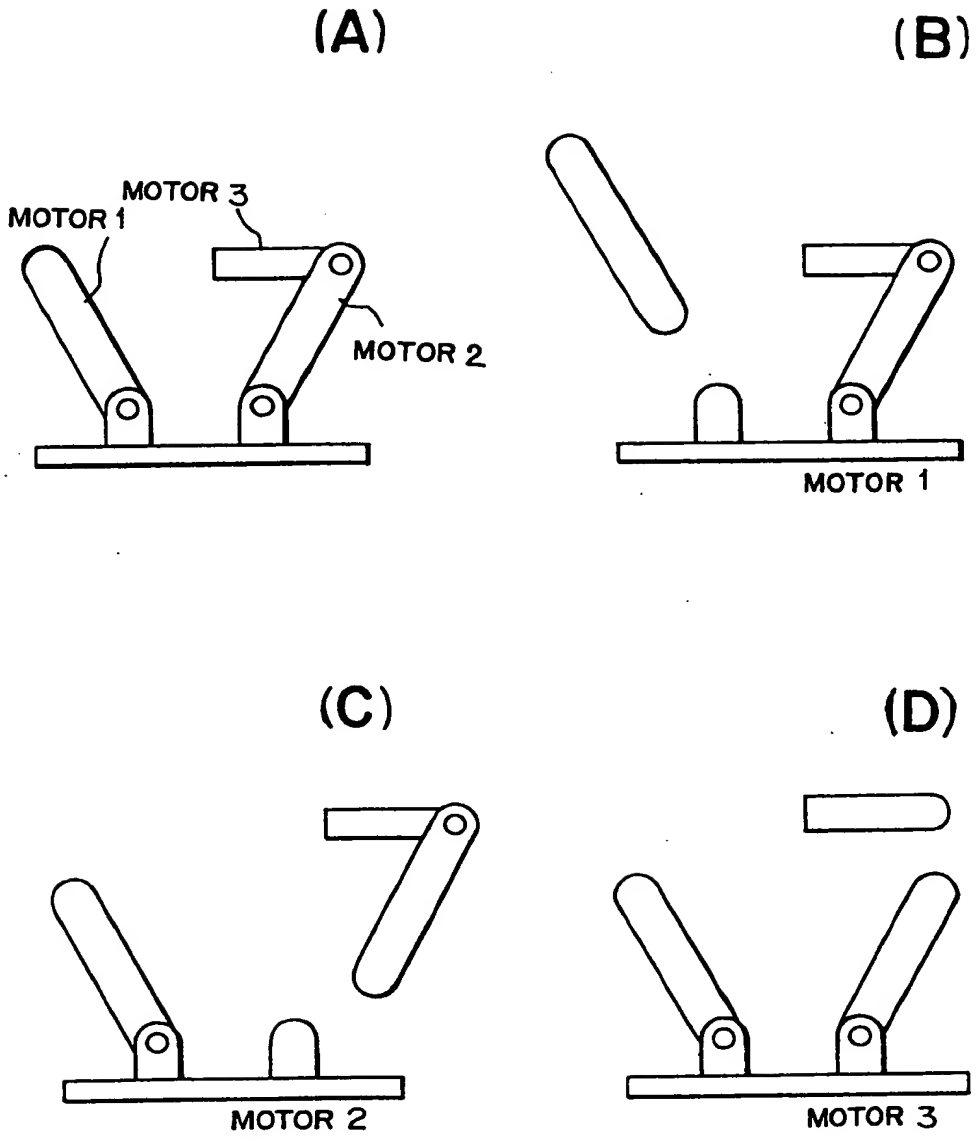
[FIG. 25]



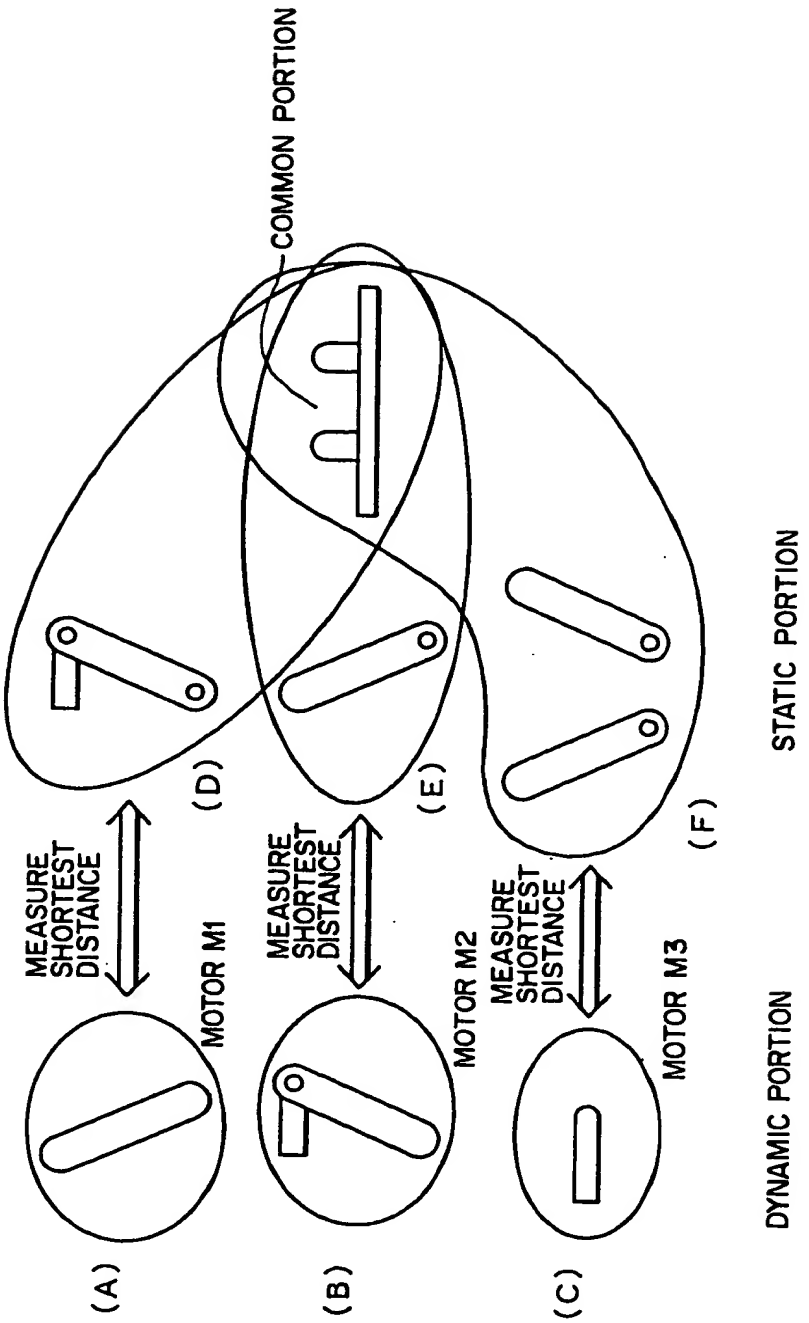
[FIG. 26]



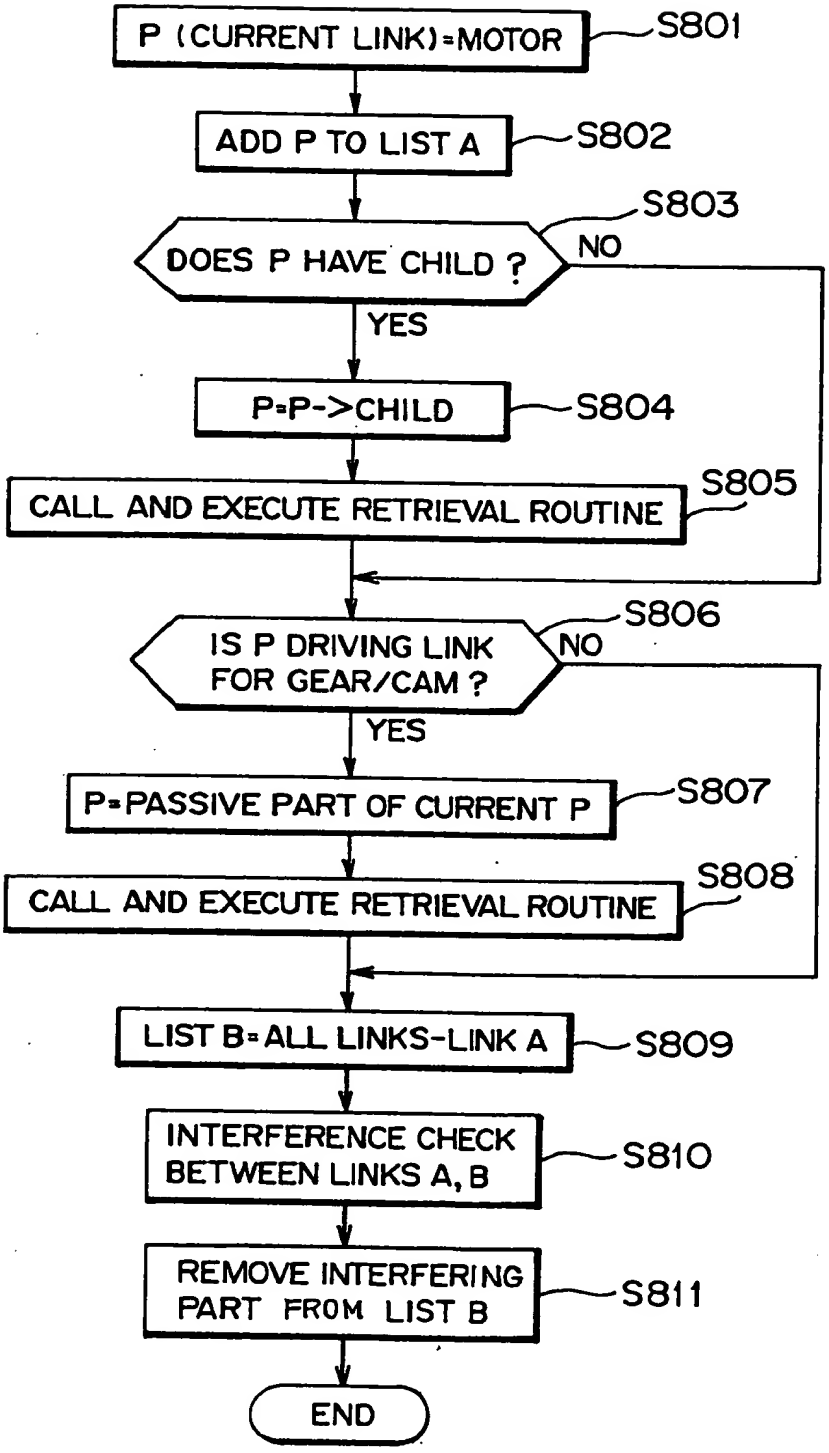
[FIG. 27]



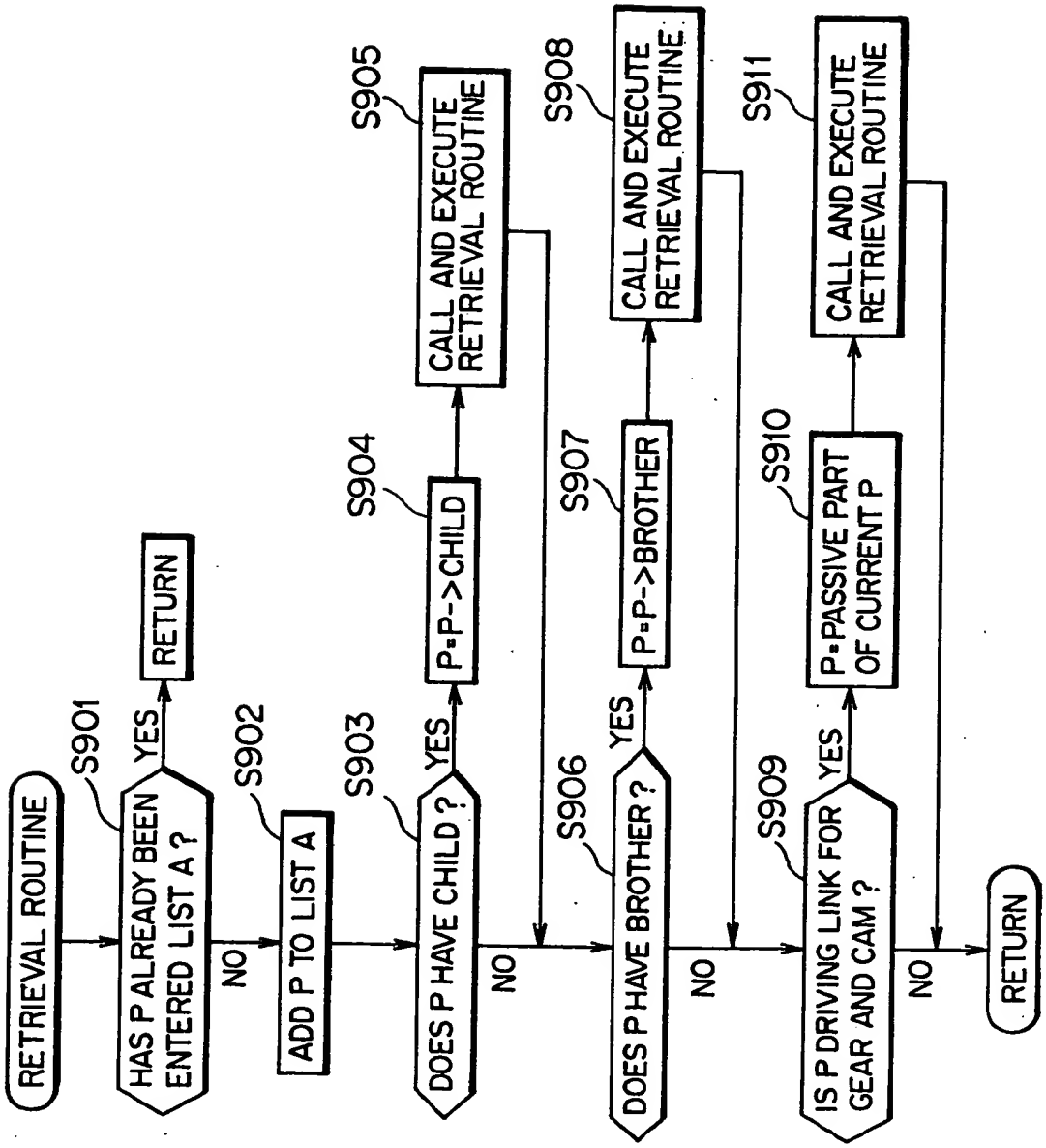
[FIG. 28]


COMMON PORTION
MEASURE SHORTEST DISTANCE
MOTOR M1
MEASURE SHORTEST DISTANCE
MOTOR M2
MEASURE SHORTEST DISTANCE
MOTOR M3
(A)
(B)
(C)
(D)
(E)
(F)
DYNAMIC PORTION
STATIC PORTION

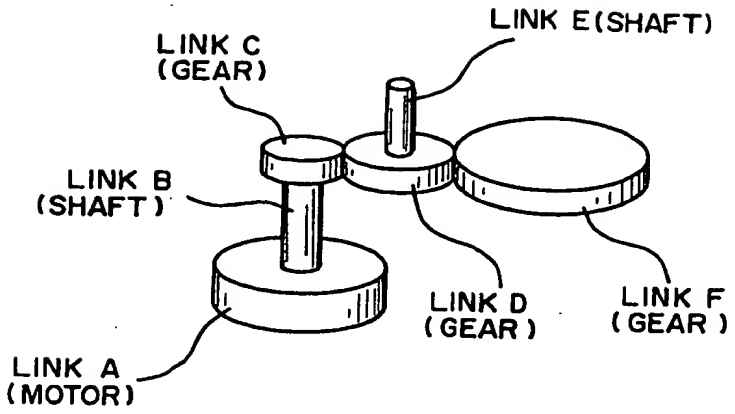
[FIG. 29]


P (CURRENT LINK)=MOTOR
S801
ADD P TO LIST A
S802
DOES P HAVE CHILD ?
S803
NO
YES
P=P->CHILD
S804
CALL AND EXECUTE RETRIEVAL ROUTINE
S805
IS P DRIVING LINK FOR GEAR/CAM ?
S806
NO
YES
P=PASSIVE PART OF CURRENT P
S807
CALL AND EXECUTE RETRIEVAL ROUTINE
S808
LIST B=ALL LINKS-LINK A
S809
INTERFERENCE CHECK BETWEEN LINKS A, B
S810
REMOVE INTERFERING PART FROM LIST B
S811
END

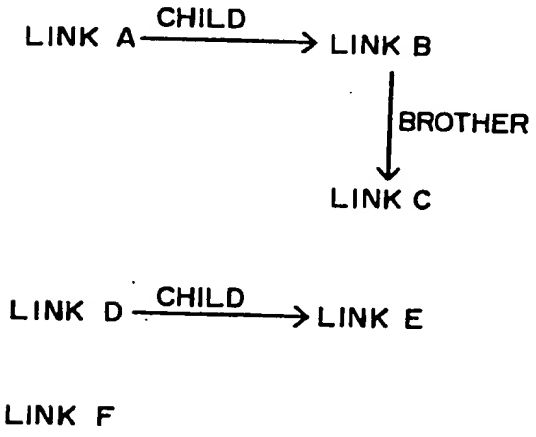
[FIG. 30]


RETRIEVAL ROUTINE
S901
HAS P ALREADY BEEN ENTERED LIST A ?
YES
RETURN
NO
S902
ADD P TO LIST A
S903
DOES P HAVE CHILD ?
YES
S904
P=P->CHILD
S905
CALL AND EXECUTE RETRIEVAL ROUTINE
NO
S906
DOES P HAVE BROTHER ?
YES
S907
P=P->BROTHER
S908
CALL AND EXECUTE RETRIEVAL ROUTINE
NO
S909
IS P DRIVING LINK FOR GEAR AND CAM ?
YES
S910
P=PASSIVE PART OF CURRENT P
S911
CALL AND EXECUTE RETRIEVAL ROUTINE
NO
RETURN

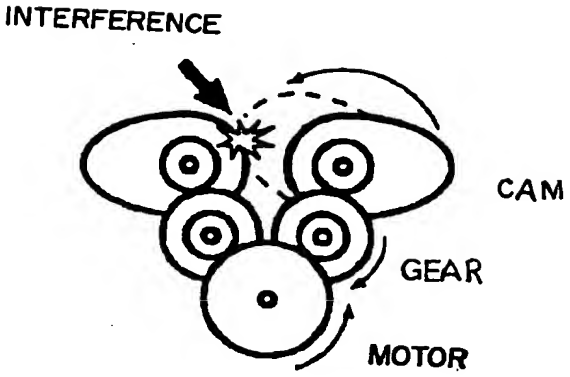
[FIG. 31]


LINK E(SHAFT)
LINK C
(GEAR)
LINK B
(SHAFT)
LINK D
(GEAR)
LINK F
(GEAR)
LINK A
(MOTOR)

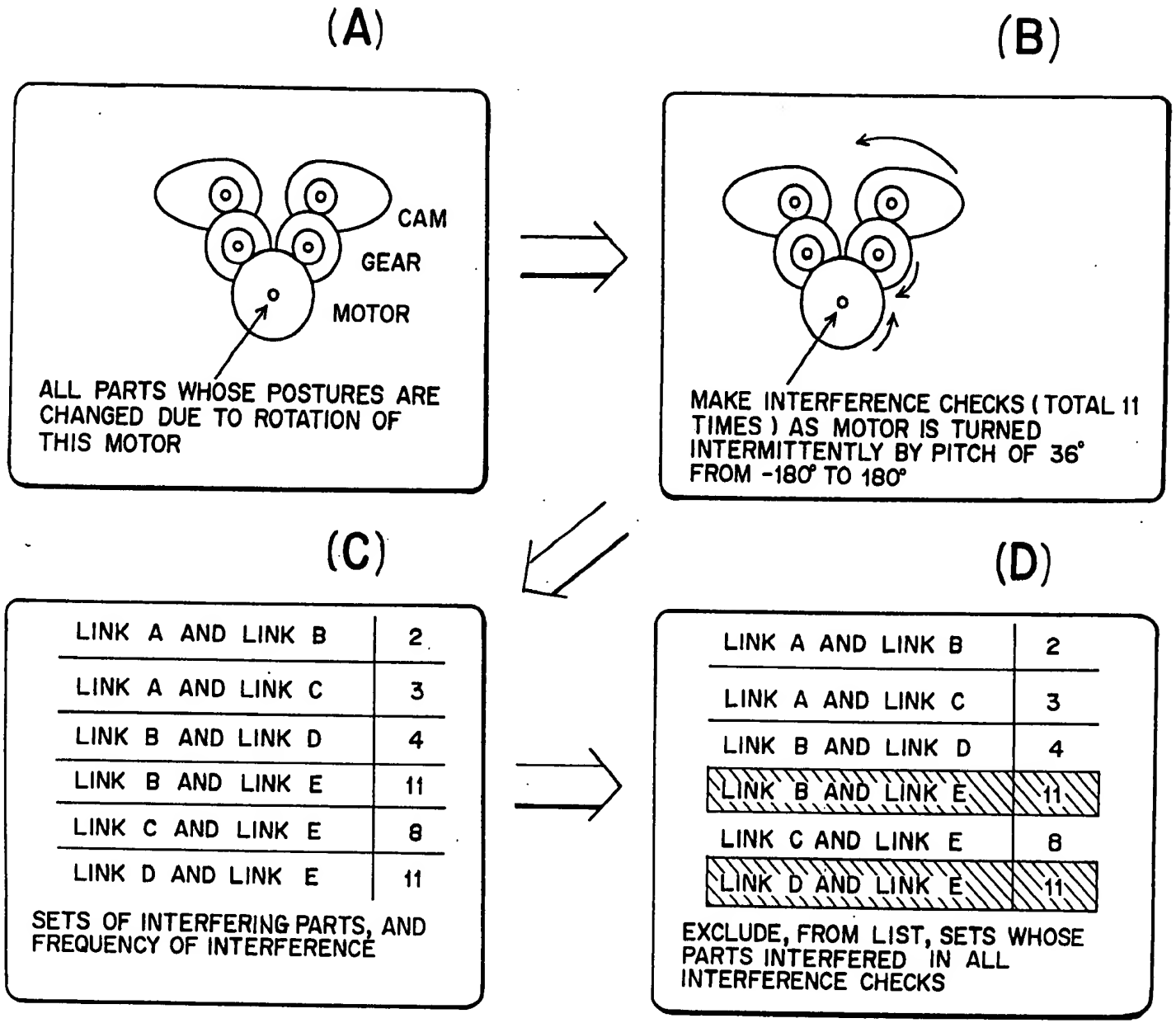
[FIG. 32]


LINK A
CHILD
LINK B
BROTHER
LINK C
LINK D
CHILD
LINK E
LINK F

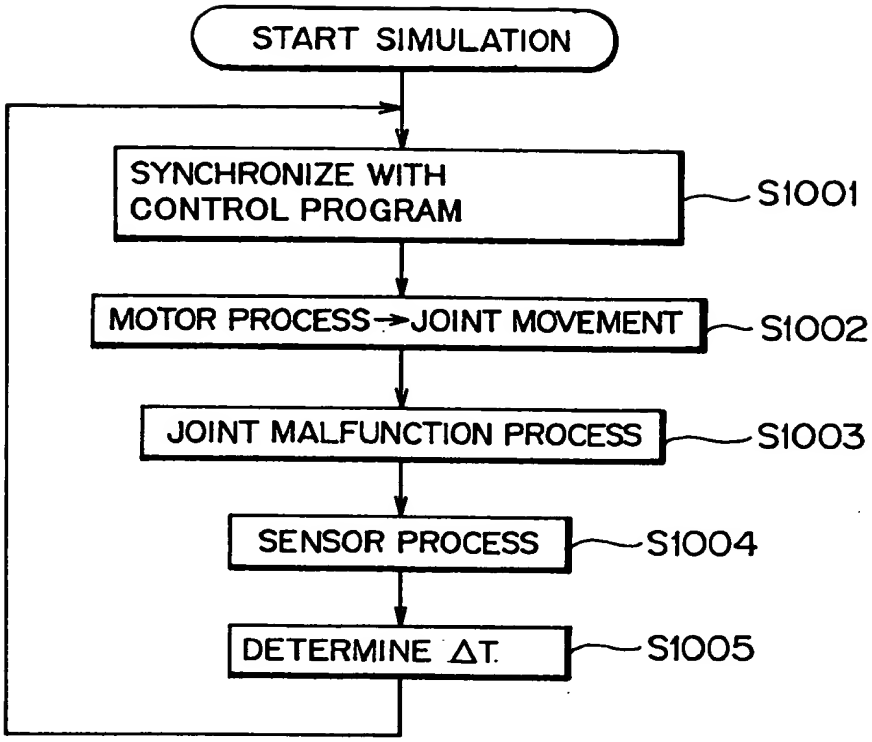
[FIG. 33]



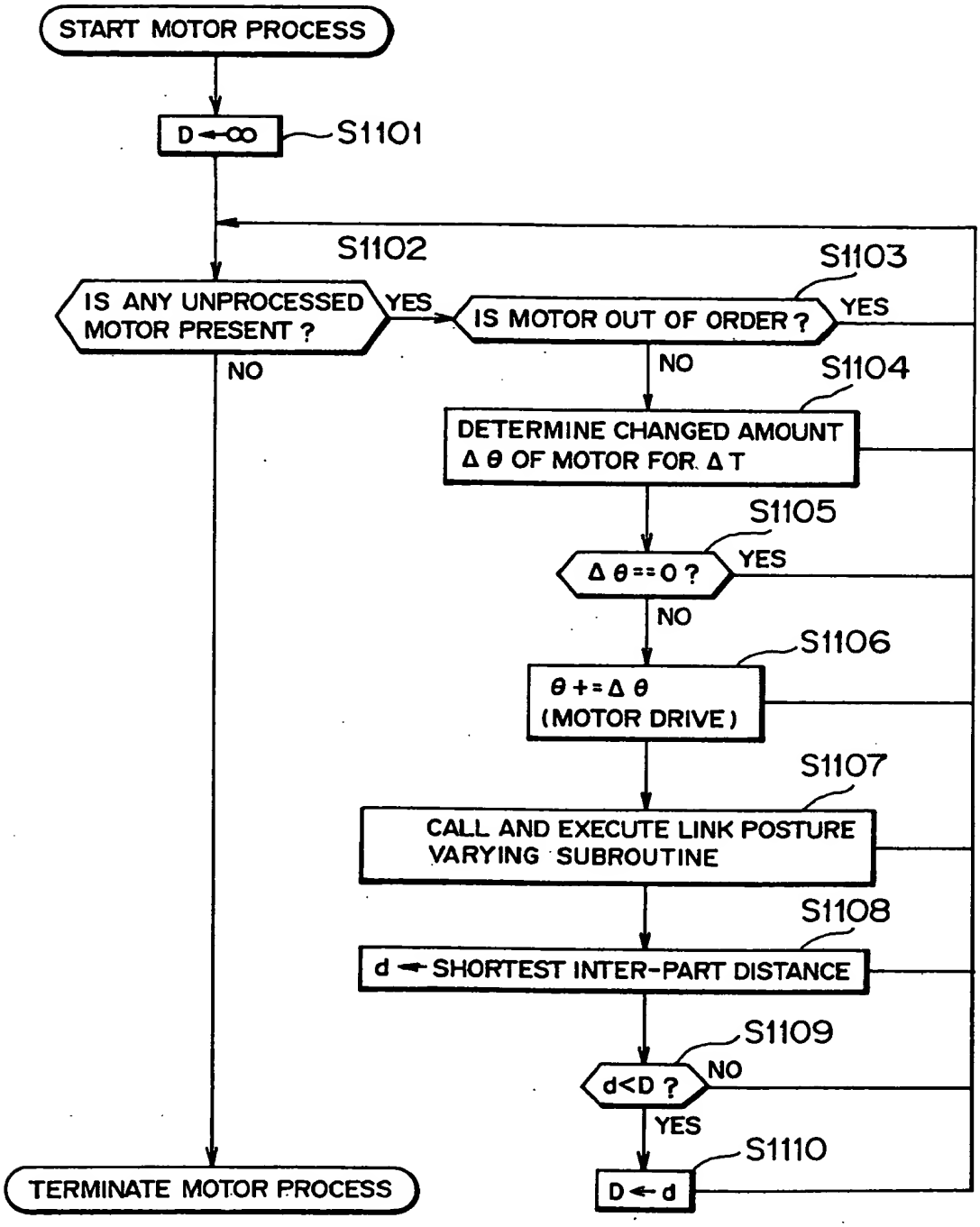
[FIG. 34]


(A)
CAM
GEAR
MOTOR
ALL PARTS WHOSE POSTURES ARE CHANGED DUE TO ROTATION OF THIS MOTOR
(B)
MAKE INTERFERENCE CHECKS (TOTAL 11 TIMES) AS MOTOR IS TURNED INTERMITTENTLY BY PITCH OF 36° FROM -180° TO 180°
(C)
LINK A AND LINK B 2
LINK A AND LINK C 3
LINK B AND LINK D 4
LINK B AND LINK E 11
LINK C AND LINK E 8
LINK D AND LINK E 11
SETS OF INTERFERING PARTS, AND FREQUENCY OF INTERFERENCE
(D)
LINK A AND LINK B 2
LINK A AND LINK C 3
LINK B AND LINK D 4
LINK B AND LINK E 11
LINK C AND LINK E 8
LINK D AND LINK E 11
EXCLUDE, FROM LIST, SETS WHOSE PARTS INTERFERED IN ALL INTERFERENCE CHECKS

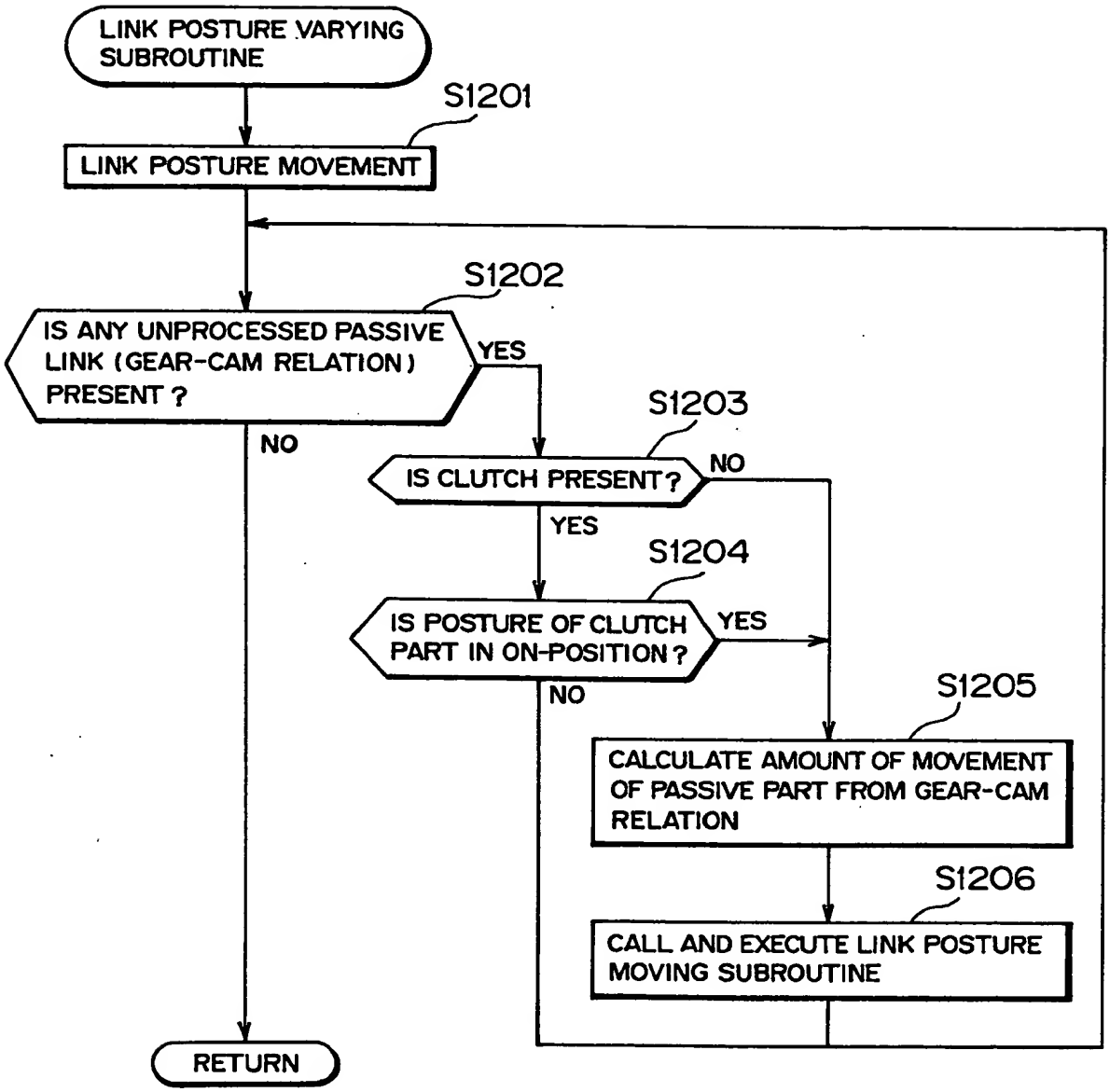
[FIG. 35]


START SIMULATION
SYNCHRONIZE WITH CONTROL PROGRAM
S1001
MOTOR PROCESS→JOINT MOVEMENT
S1002
JOINT MALFUNCTION PROCESS
S1003
SENSOR PROCESS
S1004
DETERMINE ΔT
S1005

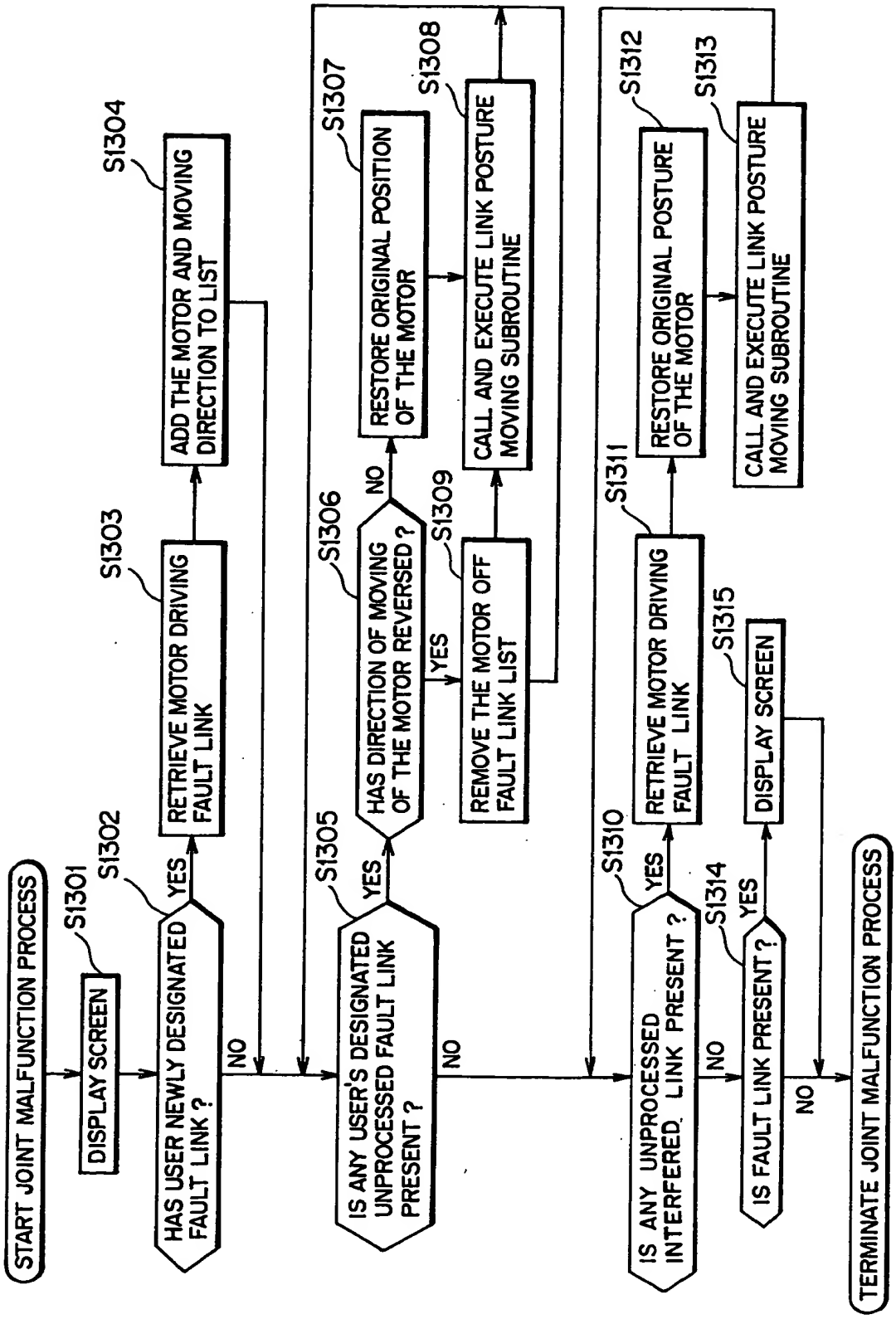
[FIG. 36]


START MOTOR PROCESS
D←∞
S1101
S1102
IS ANY UNPROCESSED MOTOR PRESENT ?
YES
NO
S1103
IS MOTOR OUT OF ORDER ?
YES
NO
S1104
DETERMINE CHANGED AMOUNT Δθ OF MOTOR FOR Δ T
S1105
Δθ==0 ?
YES
NO
S1106
θ+=Δθ (MOTOR DRIVE)
S1107
CALL AND EXECUTE LINK POSTURE VARYING SUBROUTINE
S1108
d ← SHORTEST INTER-PART DISTANCE
S1109
d<D ?
NO
YES
S1110
D←d
TERMINATE MOTOR PROCESS

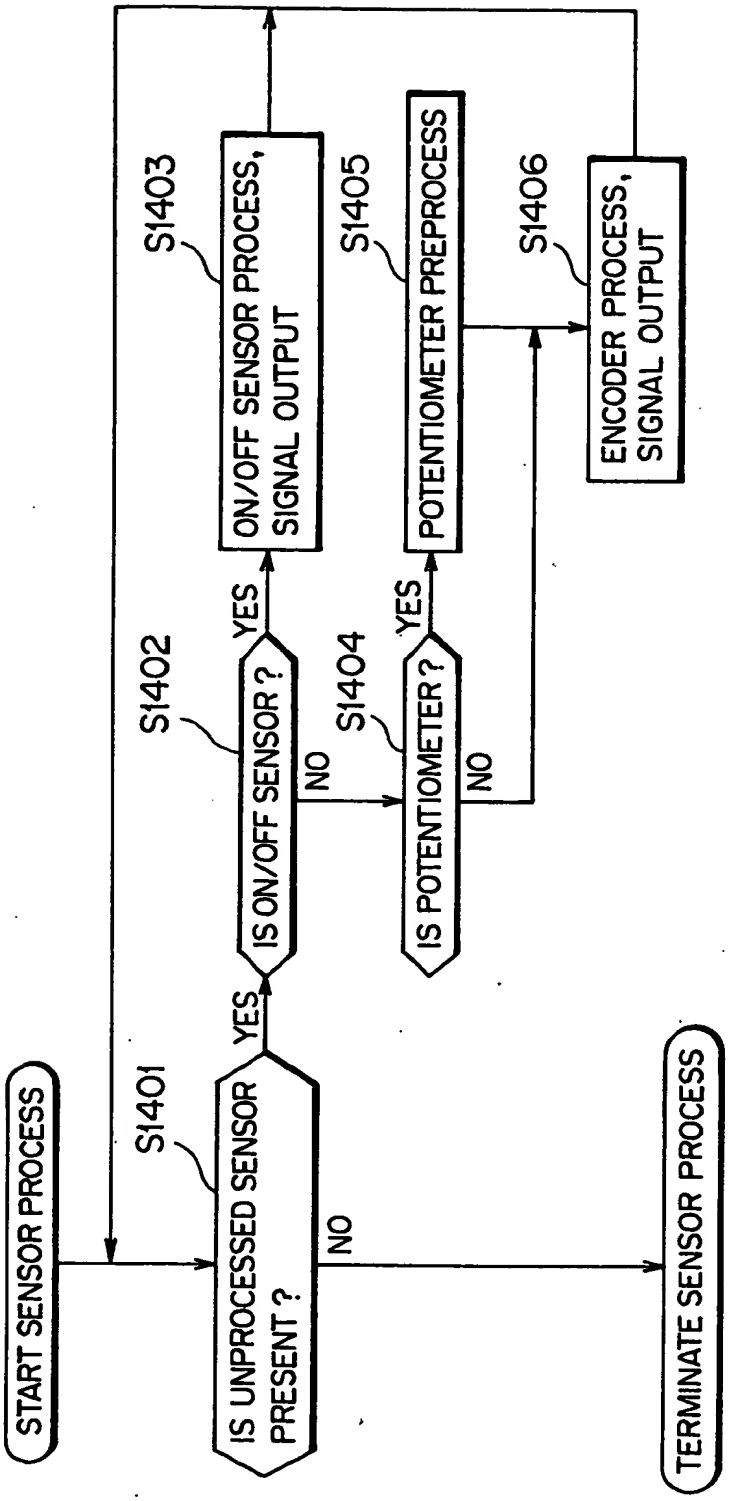
[FIG. 37]


LINK POSTURE VARYING SUBROUTINE
S1201
LINK POSTURE MOVEMENT
S1202
IS ANY UNPROCESSED PASSIVE LINK (GEAR-CAM RELATION) PRESENT ?
YES
NO
S1203
IS CLUTCH PRESENT?
NO
YES
S1204
IS POSTURE OF CLUTCH PART IN ON-POSITION?
YES
NO
S1205
CALCULATE AMOUNT OF MOVEMENT OF PASSIVE PART FROM GEAR-CAM RELATION
S1206
CALL AND EXECUTE LINK POSTURE MOVING SUBROUTINE
RETURN

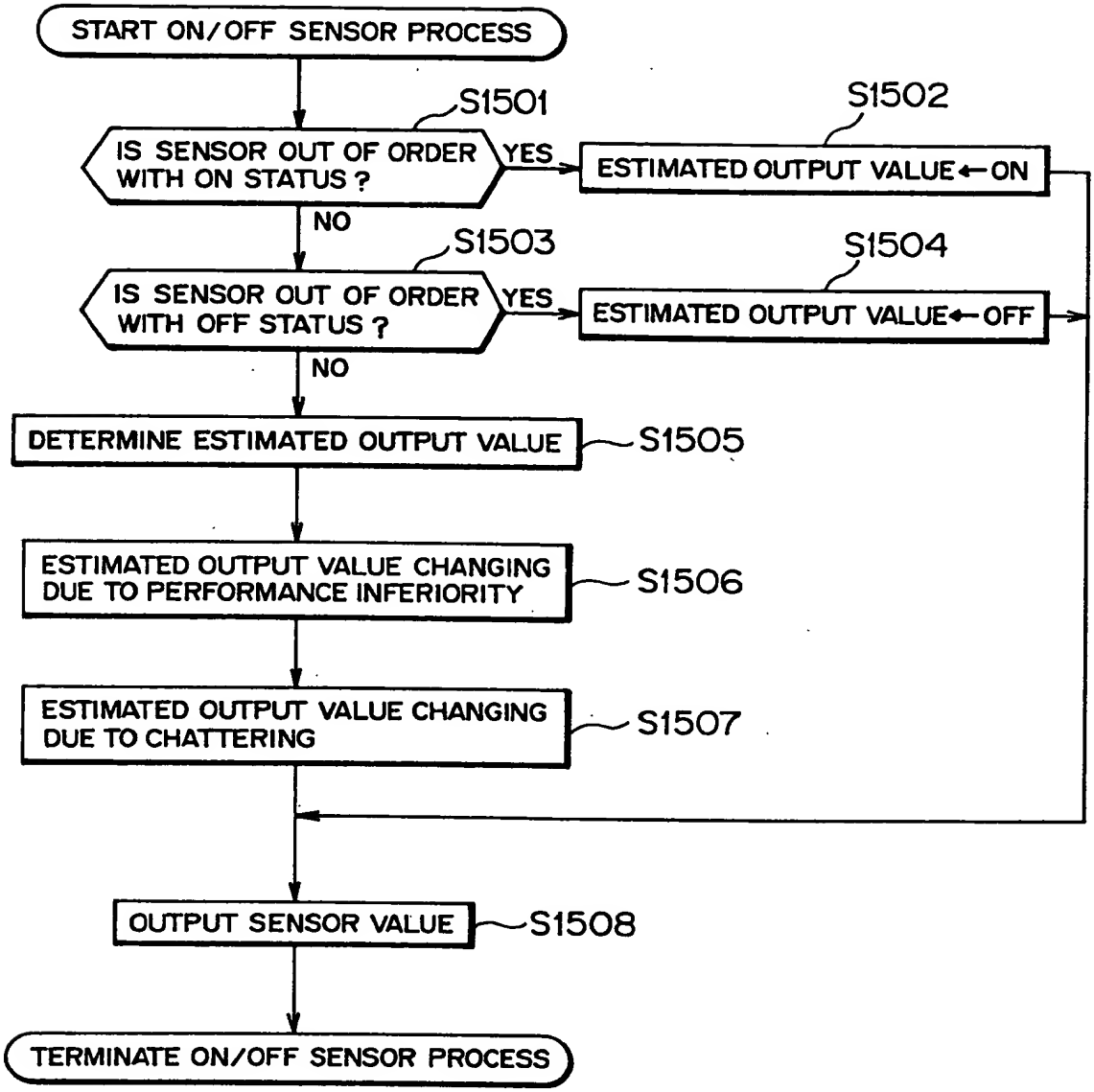
[FIG. 38]


START JOINT MALFUNCTION PROCESS
S1301
DISPLAY SCREEN
S1302
HAS USER NEWLY DESIGNATED FAULT LINK ?
YES
NO
S1303
RETRIEVE MOTOR DRIVING FAULT LINK
S1304
ADD THE MOTOR AND MOVING DIRECTION TO LIST
S1305
IS ANY USER'S DESIGNATED UNPROCESSED FAULT LINK PRESENT ?
YES
NO
S1306
HAS DIRECTION OF MOVING OF THE MOTOR REVERSED ?
YES
NO
S1307
RESTORE ORIGINAL POSITION OF THE MOTOR
S1308
CALL AND EXECUTE LINK POSTURE MOVING SUBROUTINE
S1309
REMOVE THE MOTOR OFF FAULT LINK LIST
S1310
IS ANY UNPROCESSED INTERFERED LINK PRESENT ?
YES
NO
S1311
RETRIEVE MOTOR DRIVING FAULT LINK
S1312
RESTORE ORIGINAL POSTURE OF THE MOTOR
S1313
CALL AND EXECUTE LINK POSTURE MOVING SUBROUTINE
S1314
IS FAULT LINK PRESENT ?
YES
NO
S1315
DISPLAY SCREEN
TERMINATE JOINT MALFUNCTION PROCESS

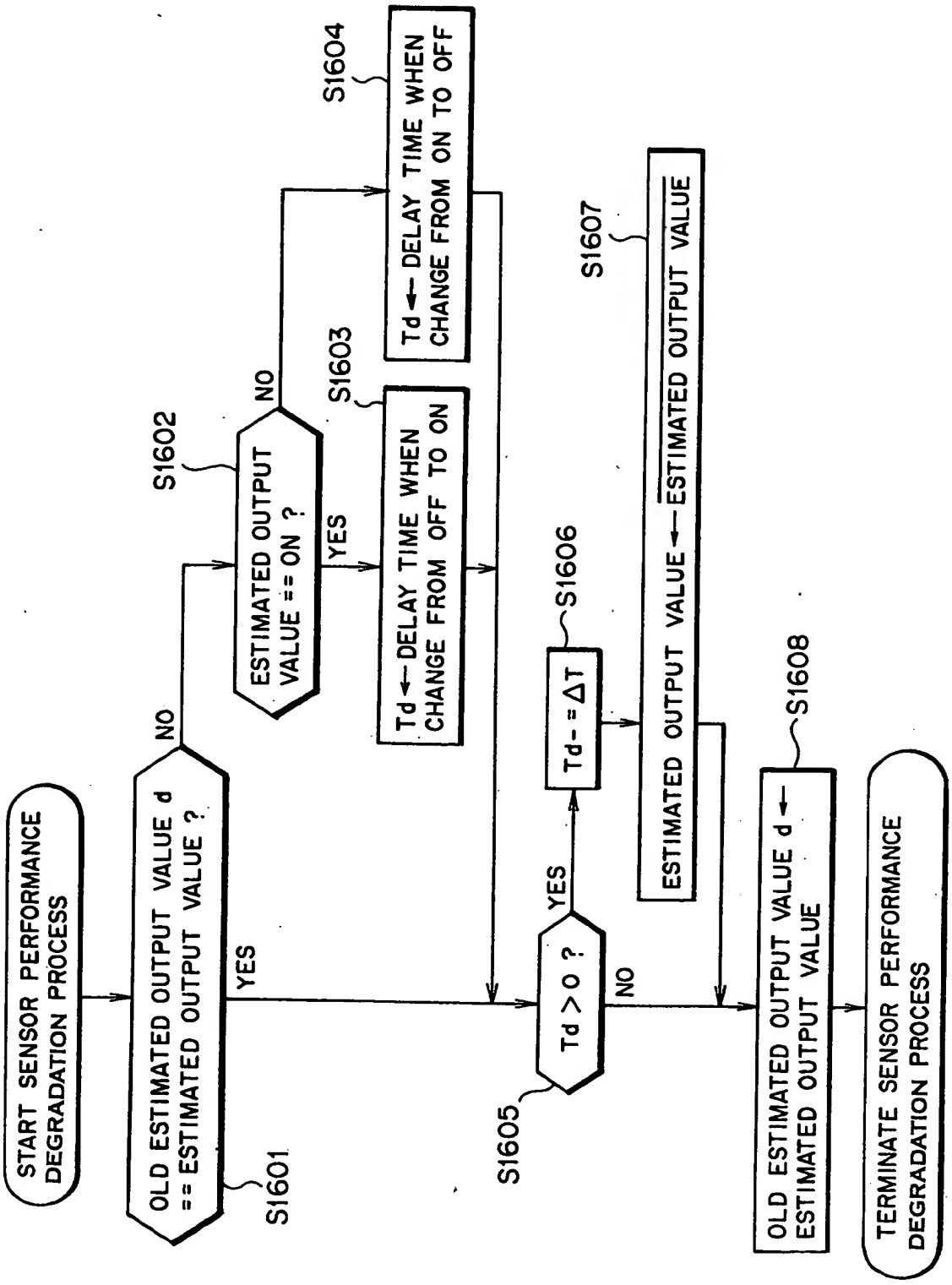
[FIG. 39]


START SENSOR PROCESS
S1401
IS UNPROCESSED SENSOR PRESENT ?
YES
NO
S1402
IS ON/OFF SENSOR ?
YES
NO
S1403
ON/OFF SENSOR PROCESS, SIGNAL OUTPUT
S1404
IS POTENTIOMETER ?
YES
NO
S1405
POTENTIOMETER PREPROCESS
S1406
ENCODER PROCESS, SIGNAL OUTPUT
TERMINATE SENSOR PROCESS

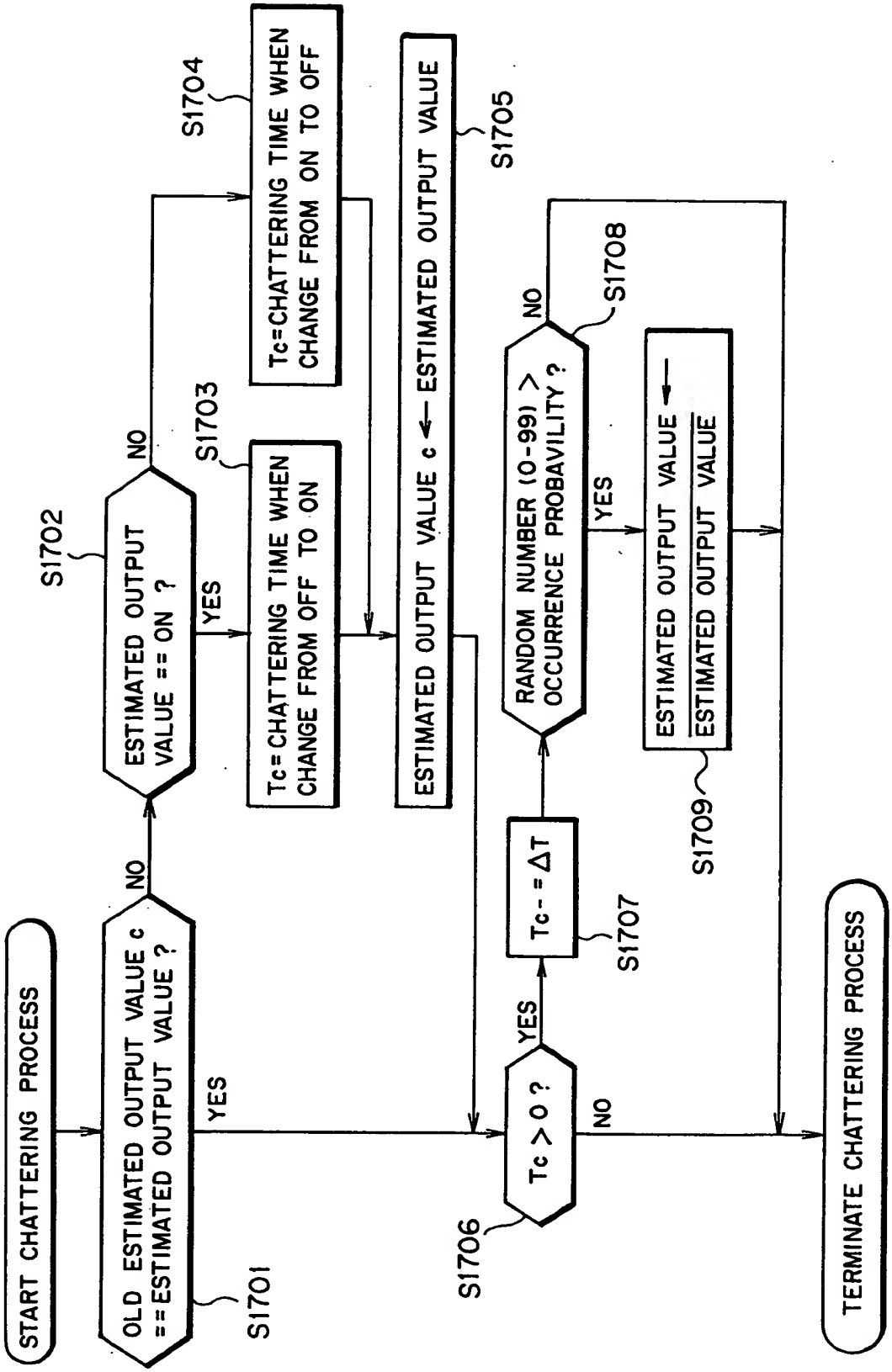
[FIG. 40]


START ON/OFF SENSOR PROCESS
S1501
IS SENSOR OUT OF ORDER WITH ON STATUS ?
YES
S1502
ESTIMATED OUTPUT VALUE ← ON
NO
S1503
IS SENSOR OUT OF ORDER WITH OFF STATUS ?
YES
S1504
ESTIMATED OUTPUT VALUE ← OFF
NO
DETERMINE ESTIMATED OUTPUT VALUE
S1505
ESTIMATED OUTPUT VALUE CHANGING DUE TO PERFORMANCE INFERIORITY
S1506
ESTIMATED OUTPUT VALUE CHANGING DUE TO CHATTERING
S1507
OUTPUT SENSOR VALUE
S1508
TERMINATE ON/OFF SENSOR PROCESS

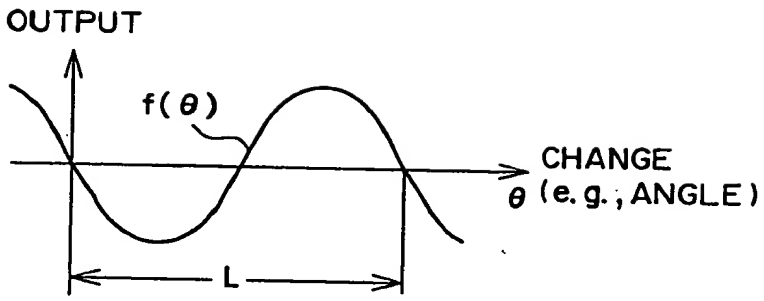
[FIG. 41]


START SENSOR PERFORMANCE DEGRADATION PROCESS
OLD ESTIMATED OUTPUT VALUE d == ESTIMATED OUTPUT VALUE ?
S1601
NO
YES
ESTIMATED OUTPUT VALUE == ON ?
S1602
NO
YES
Td ← DELAY TIME WHEN CHANGE FROM OFF TO ON
S1603
Td ← DELAY TIME WHEN CHANGE FROM ON TO OFF
S1604
Td > 0 ?
S1605
YES
NO
Td - = ΔT
S1606
ESTIMATED OUTPUT VALUE ← ESTIMATED OUTPUT VALUE
S1607
OLD ESTIMATED OUTPUT VALUE d ← ESTIMATED OUTPUT VALUE
S1608
TERMINATE SENSOR PERFORMANCE DEGRADATION PROCESS

[FIG. 42]

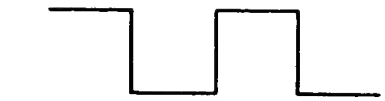

START CHATTERING PROCESS
OLD ESTIMATED OUTPUT VALUE c == ESTIMATED OUTPUT VALUE ?
S1701
NO
YES
ESTIMATED OUTPUT VALUE == ON ?
S1702
NO
YES
Tc=CHATTERING TIME WHEN CHANGE FROM OFF TO ON
S1703
Tc=CHATTERING TIME WHEN CHANGE FROM ON TO OFF
S1704
ESTIMATED OUTPUT VALUE c ← ESTIMATED OUTPUT VALUE
S1705
Tc > 0 ?
S1706
YES
NO
Tc - - = ΔT
S1707
RANDOM NUMBER (0-99) > OCCURRENCE PROBAVILITY ?
S1708
NO
YES
ESTIMATED OUTPUT VALUE ← ESTIMATED OUTPUT VALUE
S1709
TERMINATE CHATTERING PROCESS

[FIG. 43]



[FIG. 44]

(A)



RECTANGULAR WAVE

(B)

SIGN WAVE

(C)

TRIANGULAR WAVE

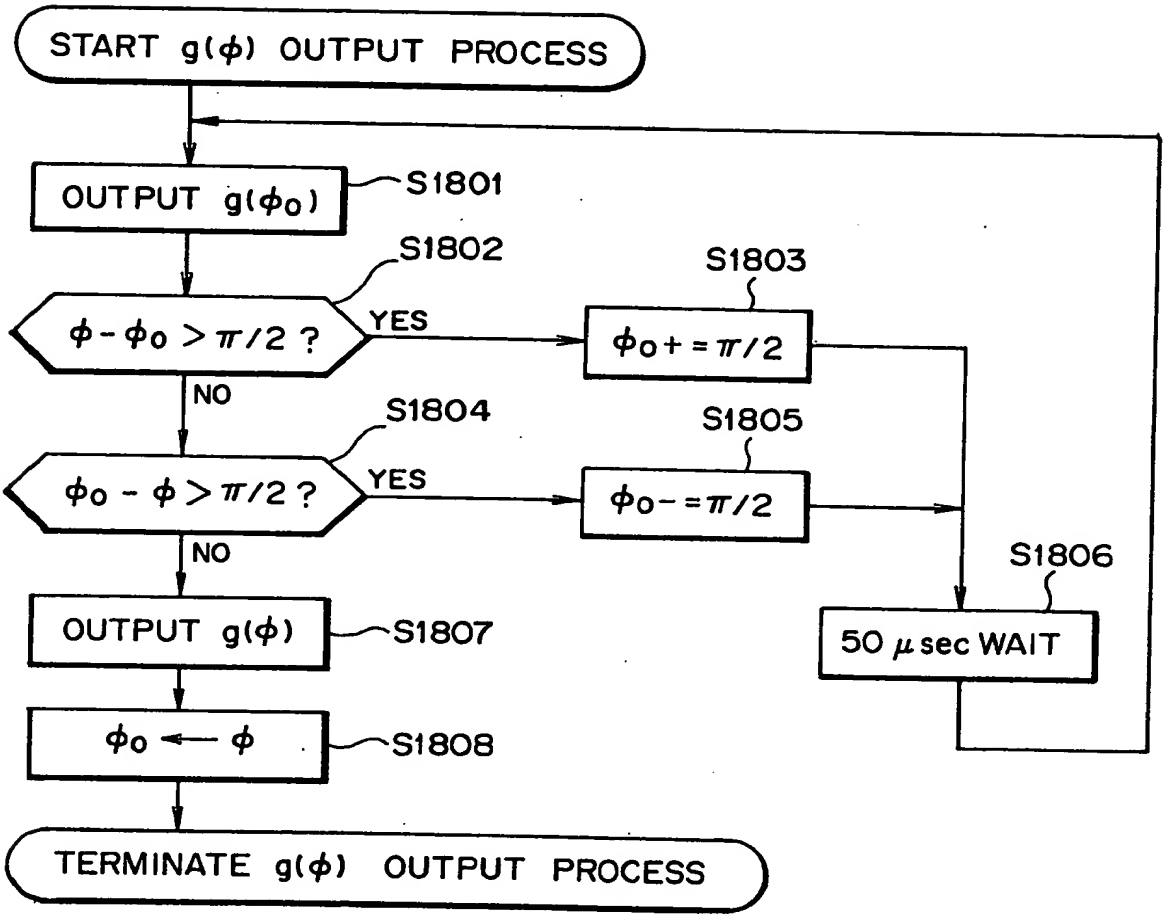
(D)

SAWTEETH WAVE (RIGHT SIDE UP)

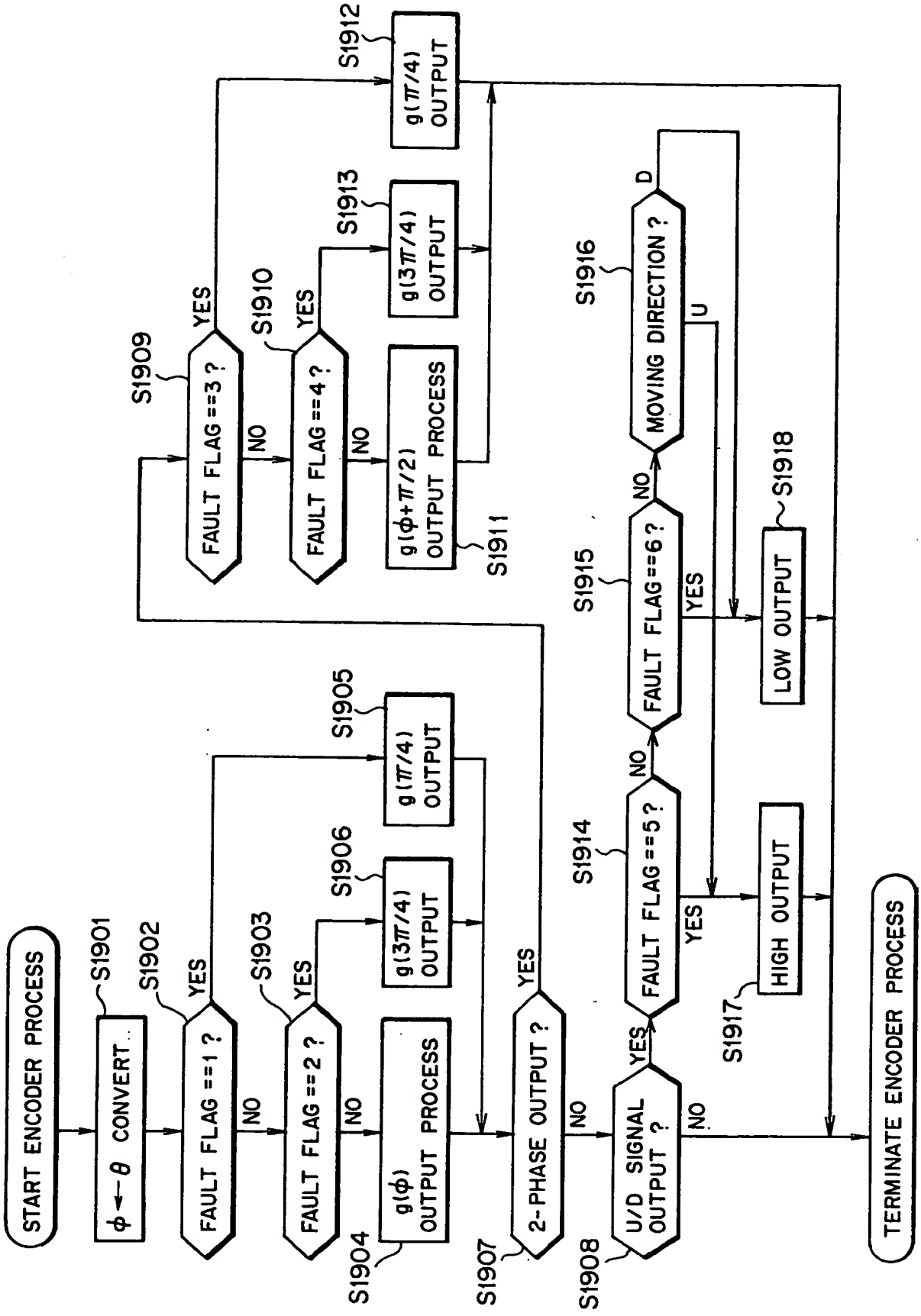
(E)

SAWTEETH WAVE (RIGHT SIDE DOWN)

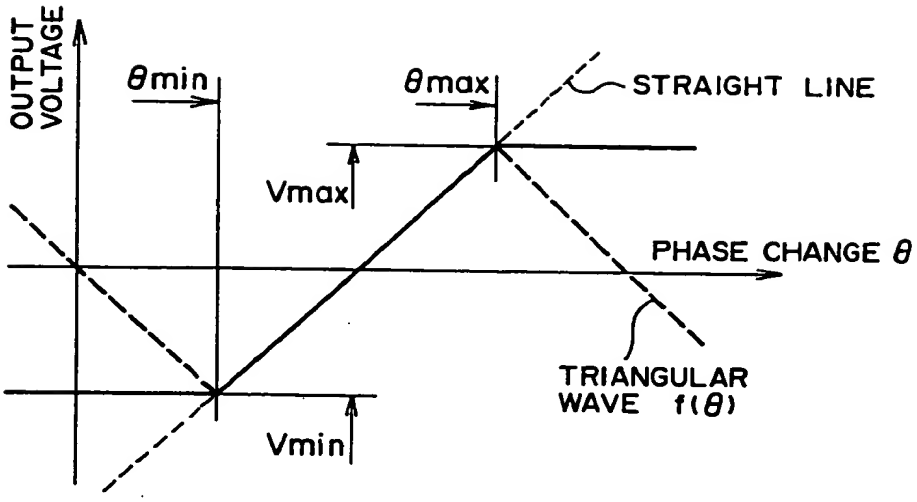
[FIG. 45]


START g(φ) OUTPUT PROCESS
OUTPUT g(φo)
S1801
S1802
φ-φo > π/2 ?
YES
S1803
φo+ = π/2
NO
S1804
φo - φ > π/2 ?
YES
S1805
φo- = π/2
NO
S1806
OUTPUT g(φ)
S1807
50 μsec WAIT
φo ← φ
S1808
TERMINATE g(φ) OUTPUT PROCESS

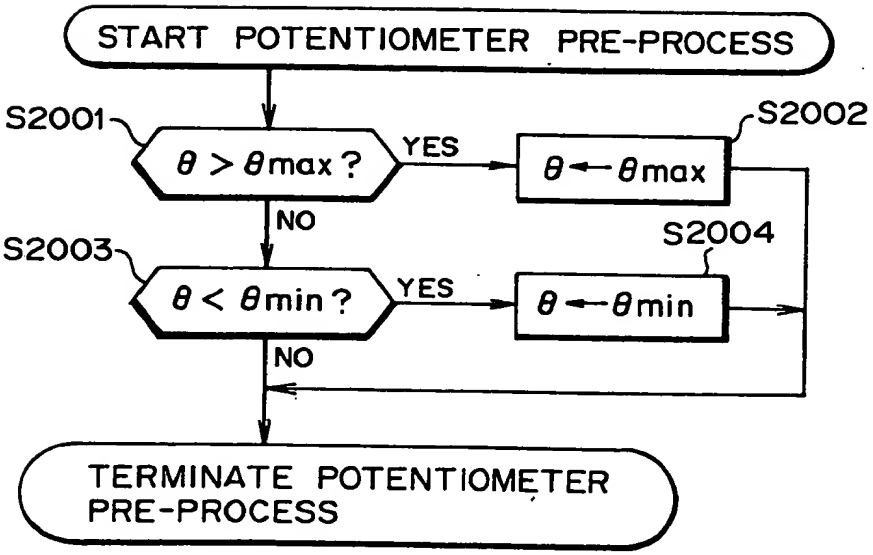
[FIG. 46]


START ENCODER PROCESS
S1901
φ ← θ CONVERT
S1902
FAULT FLAG == 1 ?
YES
NO
S1903
FAULT FLAG == 2 ?
YES
NO
S1904
g(φ) OUTPUT PROCESS
S1905
g(π/4) OUTPUT
S1906
g(3π/4) OUTPUT
S1907
2-PHASE OUTPUT ?
YES
NO
S1908
U/D SIGNAL OUTPUT ?
YES
NO
S1909
FAULT FLAG == 3 ?
YES
NO
S1910
FAULT FLAG == 4 ?
YES
NO
S1911
g(φ+π/2) OUTPUT PROCESS
S1912
g(π/4) OUTPUT
S1913
g(3π/4) OUTPUT
S1914
FAULT FLAG == 5 ?
YES
NO
S1915
FAULT FLAG == 6 ?
YES
NO
S1916
MOVING DIRECTION ?
D
U
S1917
HIGH OUTPUT
S1918
LOW OUTPUT
TERMINATE ENCODER PROCESS

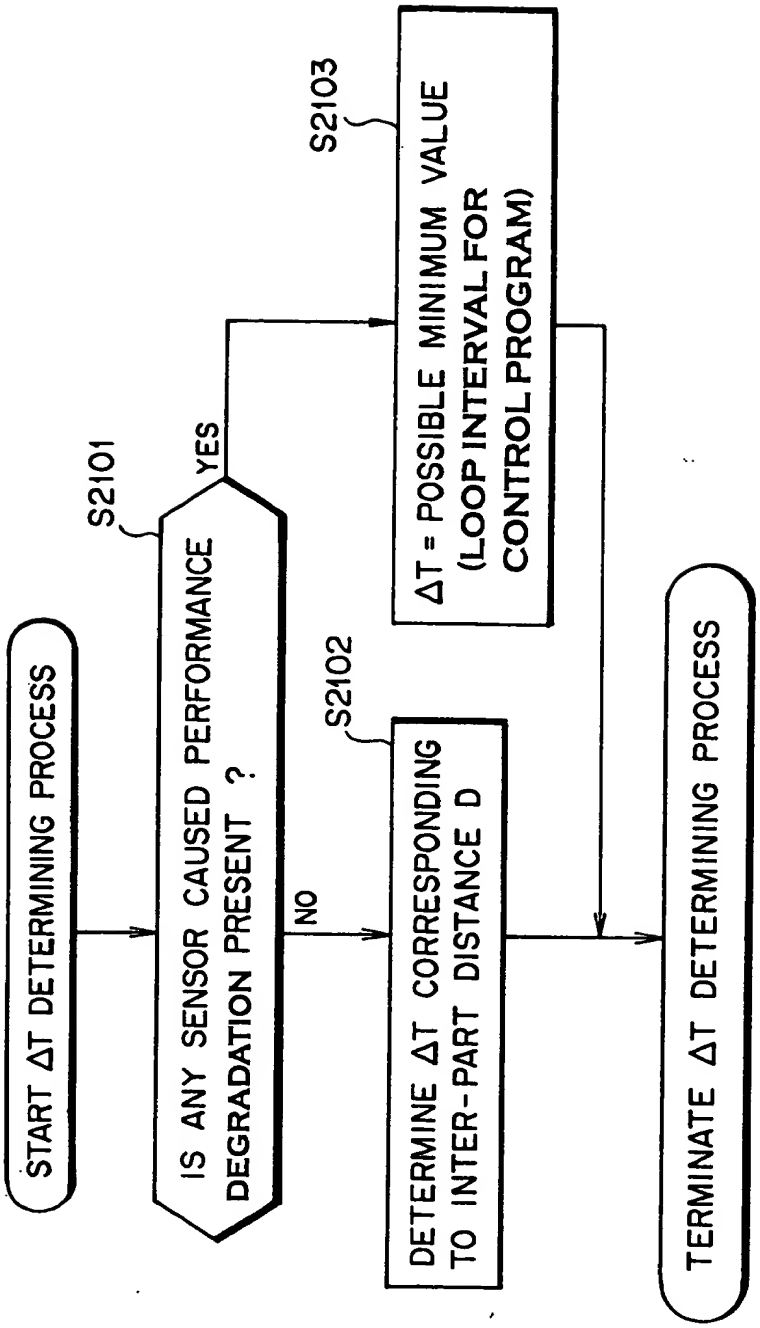
[FIG. 47]


OUTPUT VOLTAGE
θmin
θmax
STRAIGHT LINE
Vmax
PHASE CHANGE θ
TRIANGULAR WAVE f(θ)
Vmin

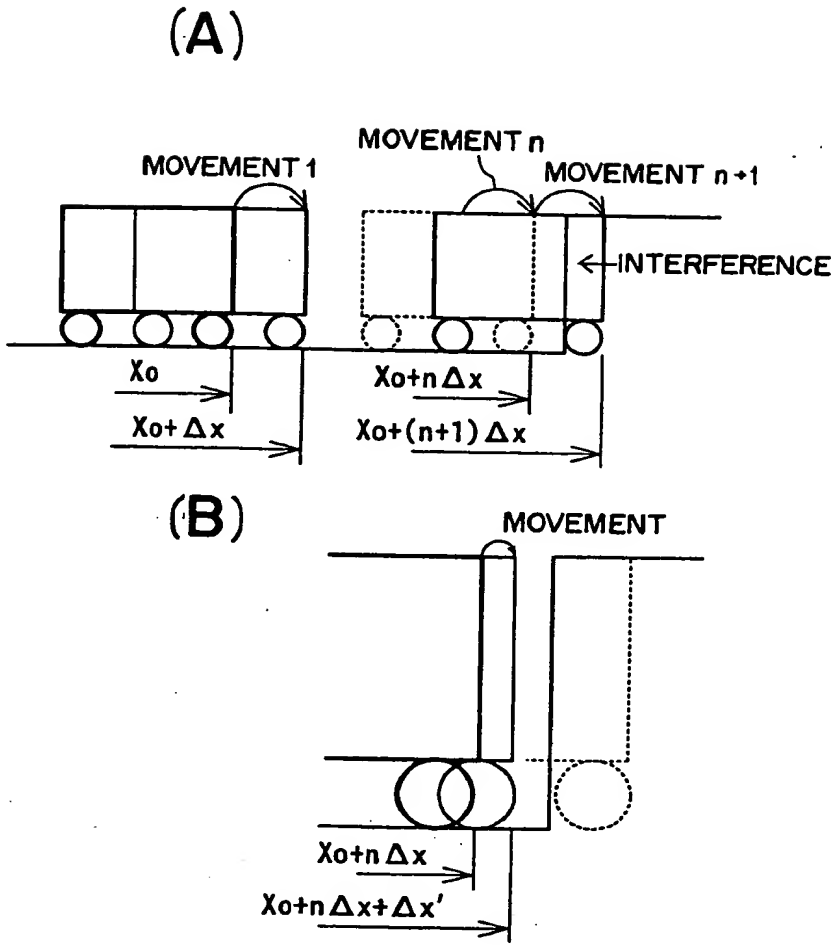
[FIG. 48]


START POTENTIOMETER PRE-PROCESS
S2001
θ > θmax ?
YES
θ ← θmax
S2002
NO
S2003
θ < θmin ?
YES
θ ← θmin
S2004
NO
TERMINATE POTENTIOMETER PRE-PROCESS

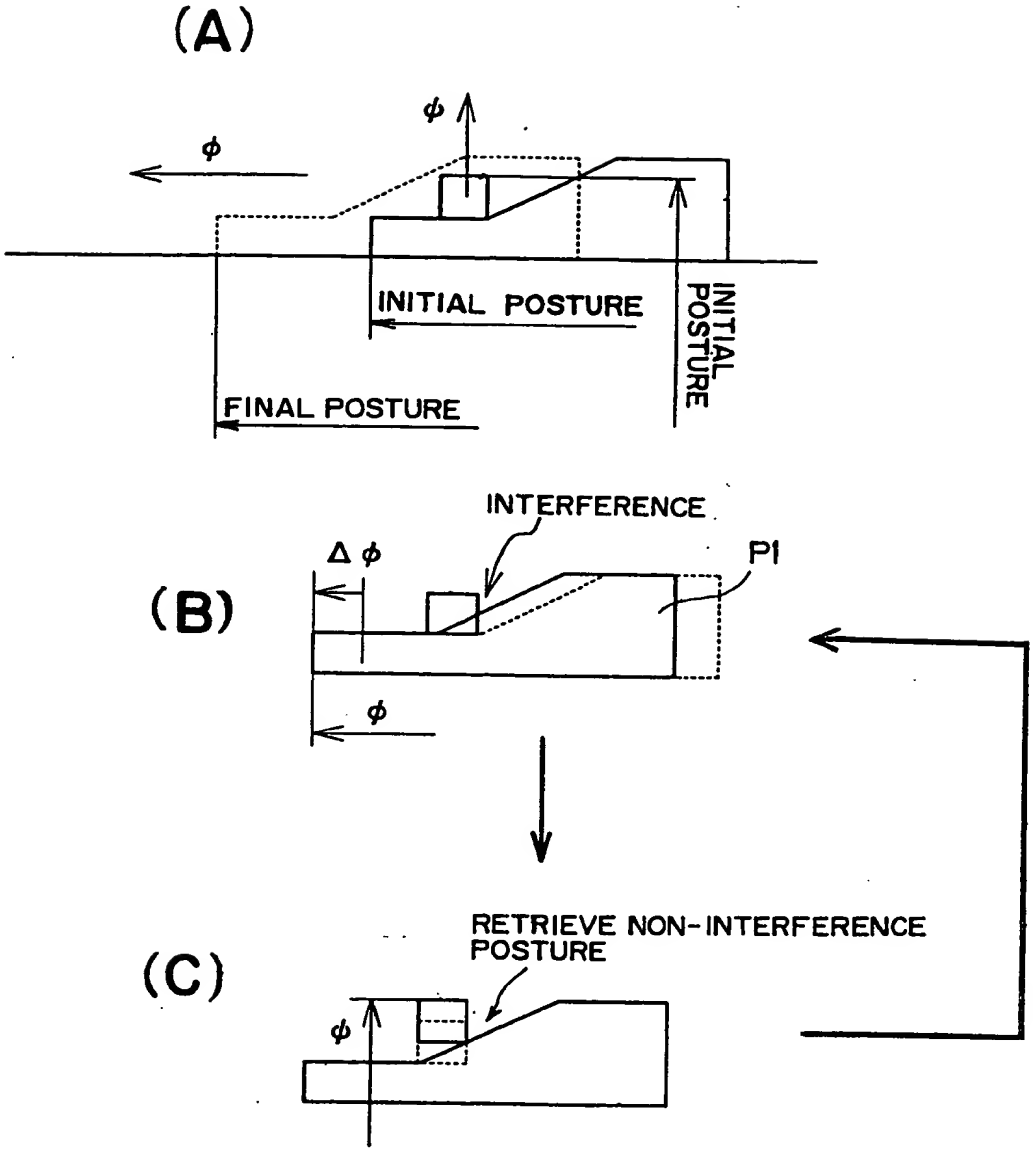
[FIG. 49]


START ΔT DETERMINING PROCESS
S2101
IS ANY SENSOR CAUSED PERFORMANCE DEGRADATION PRESENT ?
YES
NO
S2102
DETERMINE ΔT CORRESPONDING TO INTER-PART DISTANCE D
S2103
ΔT = POSSIBLE MINIMUM VALUE (LOOP INTERVAL FOR CONTROL PROGRAM)
TERMINATE ΔT DETERMINING PROCESS

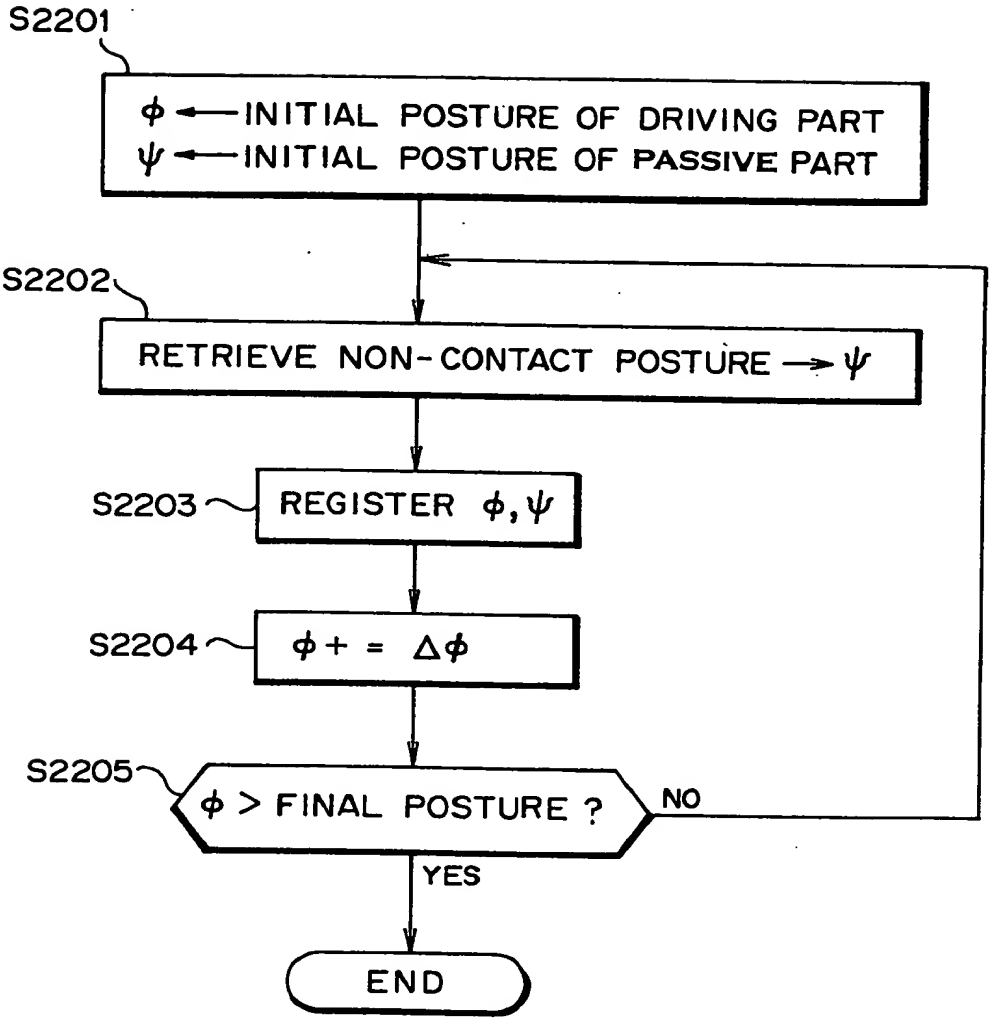
[FIG. 50]


(A)
MOVEMENT 1
MOVEMENT n
MOVEMENT n+1
INTERFERENCE
Xo
Xo+Δx
Xo+nΔx
Xo+(n+1)Δx
(B)
MOVEMENT
Xo+nΔx
Xo+nΔx+Δx'

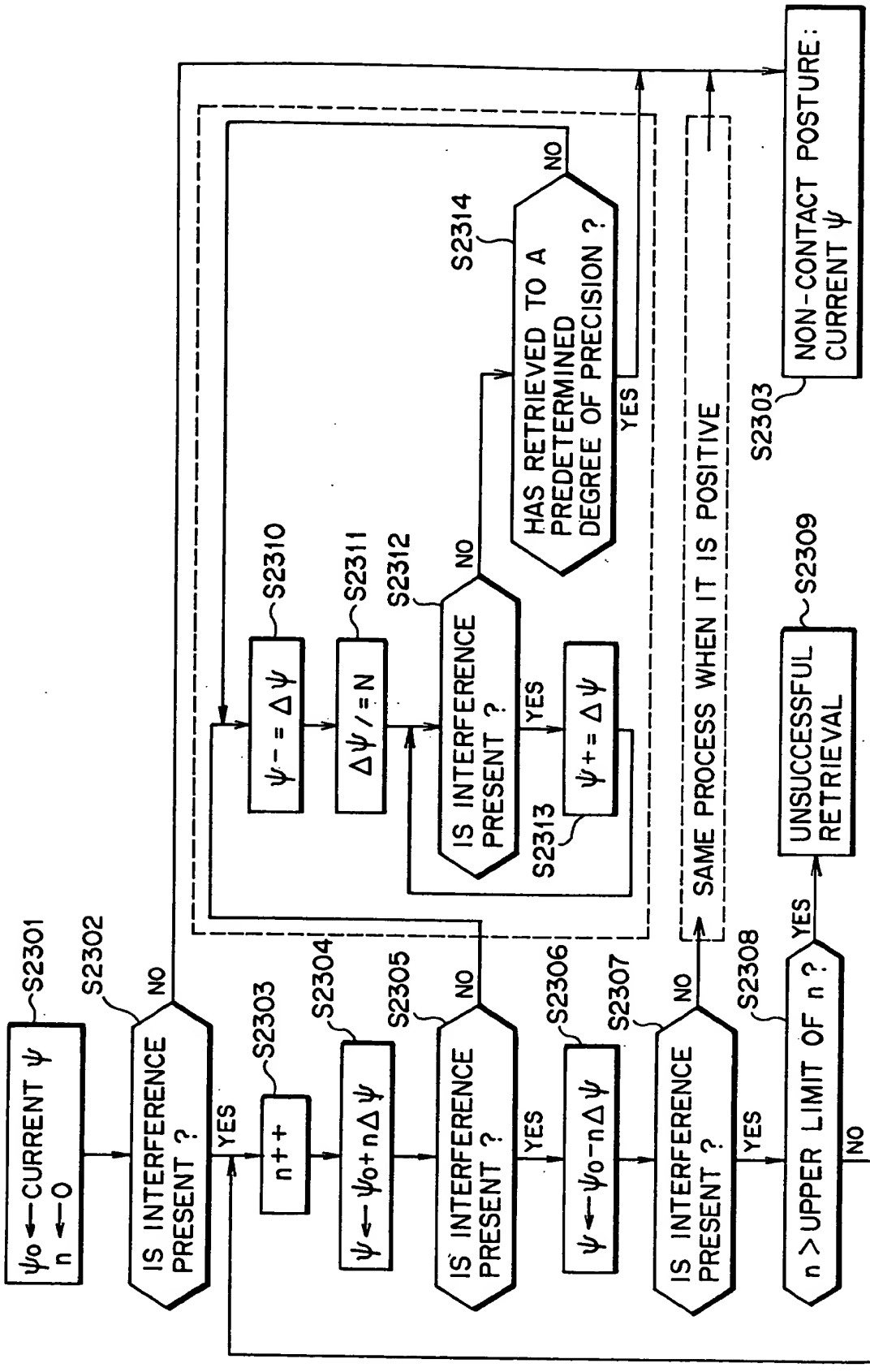
[FIG. 51]


(A)
φ
φ
INITIAL POSTURE
INITIAL POSTURE
FINAL POSTURE
(B)
INTERFERENCE
Δφ
P1
φ
(C)
RETRIEVE NON-INTERFERENCE POSTURE
φ

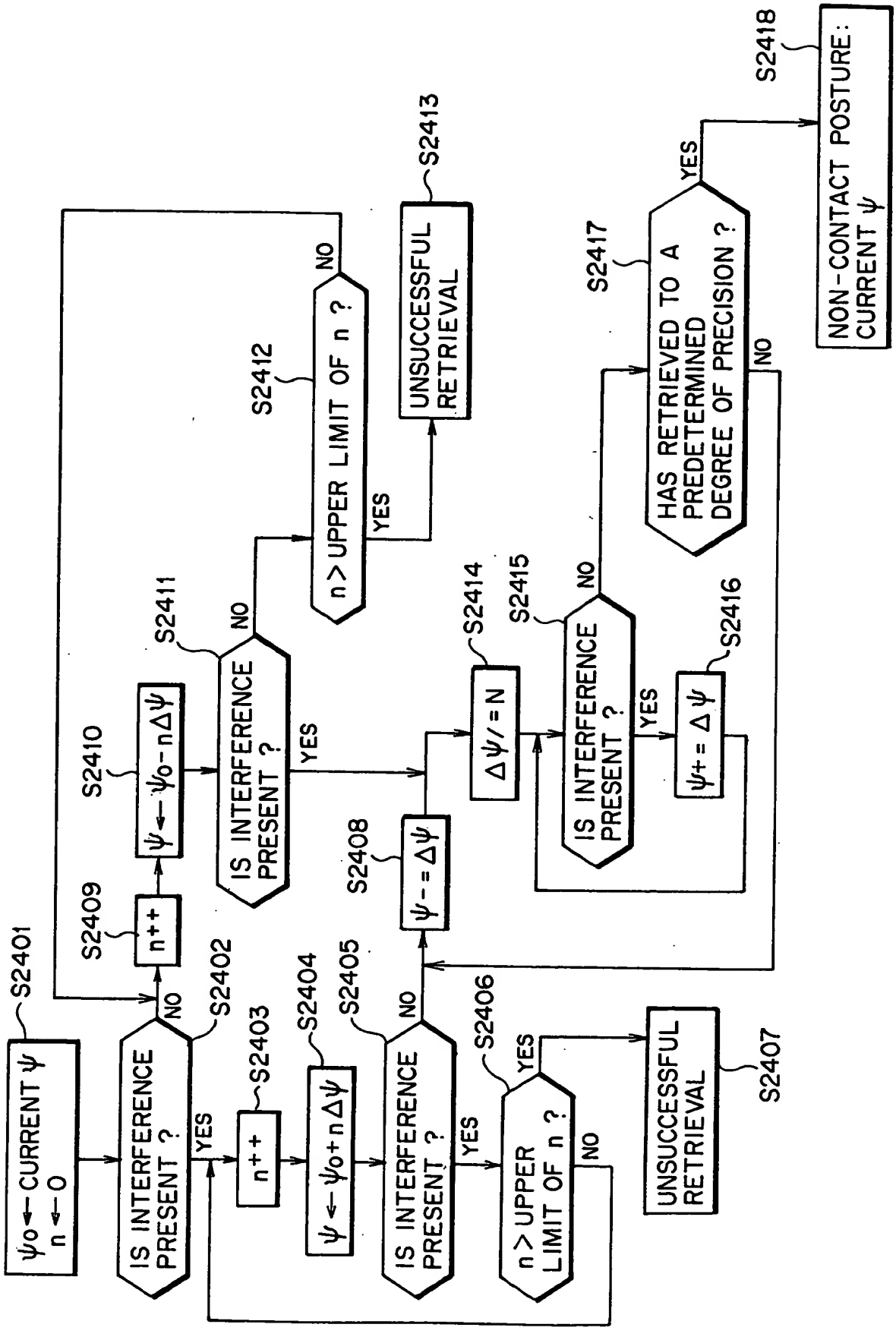
[FIG. 52]


S2201
φ ← INITIAL POSTURE OF DRIVING PART
ψ ← INITIAL POSTURE OF PASSIVE PART
S2202
RETRIEVE NON-CONTACT POSTURE → ψ
S2203
REGISTER φ, ψ
S2204
φ + = Δφ
S2205
φ > FINAL POSTURE ?
NO
YES
END

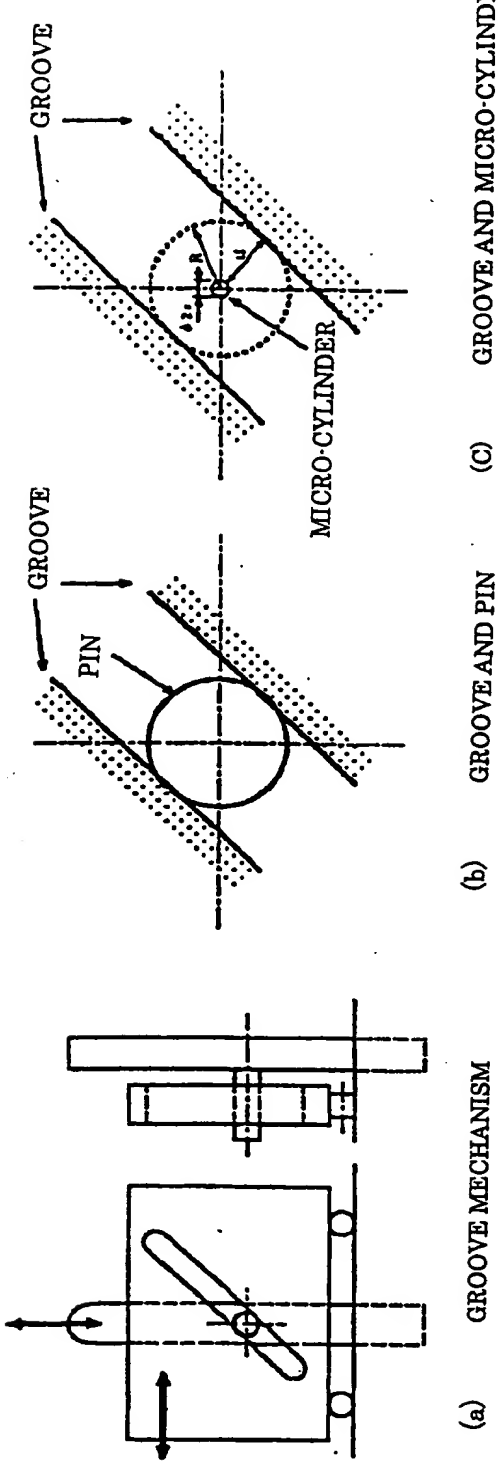
[FIG. 53]


S2301
ψo ← CURRENT ψ
n ← 0
S2302
IS INTERFERENCE PRESENT ?
NO
YES
S2303
n++
S2304
ψ ← ψo + nΔψ
S2305
IS INTERFERENCE PRESENT ?
NO
YES
S2306
ψ ← ψo − nΔψ
S2307
IS INTERFERENCE PRESENT ?
NO
YES
S2308
n > UPPER LIMIT OF n ?
YES
NO
S2309
UNSUCCESSFUL RETRIEVAL
S2303
NON-CONTACT POSTURE: CURRENT ψ
SAME PROCESS WHEN IT IS POSITIVE
S2310
ψ− = Δψ
S2311
Δψ / = N
S2312
IS INTERFERENCE PRESENT ?
NO
YES
S2313
ψ+ = Δψ
S2314
HAS RETRIEVED TO A PREDETERMINED DEGREE OF PRECISION ?
NO
YES

[FIG. 54]


S2401
ψo ← CURRENT ψ
n ← 0
S2402
IS INTERFERENCE PRESENT ?
NO
YES
S2403
n++
S2404
ψ ← ψo + nΔψ
S2405
IS INTERFERENCE PRESENT ?
NO
YES
S2406
n > UPPER LIMIT OF n ?
YES
NO
S2407
UNSUCCESSFUL RETRIEVAL
S2408
ψ- = Δψ
S2409
n++
S2410
ψ ← ψo - nΔψ
S2411
IS INTERFERENCE PRESENT ?
NO
YES
S2412
n > UPPER LIMIT OF n ?
NO
YES
S2413
UNSUCCESSFUL RETRIEVAL
S2414
Δψ/ = N
S2415
IS INTERFERENCE PRESENT ?
NO
YES
S2416
ψ+ = Δψ
S2417
HAS RETRIEVED TO A PREDETERMINED DEGREE OF PRECISION ?
YES
NO
S2418
NON-CONTACT POSTURE: CURRENT ψ

[FIG. 55]



(a) GROOVE MECHANISM (b) GROOVE AND PIN (C) GROOVE AND MICRO-CYLINDER